Fluid-Structure Interaction through multi-code coupling in HPC systems

J.C. Cajas[1], B. Eguzkitza[1], G. Houzeaux[1], M. Vázquez[1]

1) Barcelona Supercomputing Center-Centro Nacional de Supercomputación, Barcelona, Spain

Outline

- Preliminaries
- The ingredients
- FSI problems
- Results

Multiphysics strategies

Monolithic approach

The coupled problem is casted as a whole, resulting in a single algebraic (matrix) problem. The coupling conditions are part of the solution of the problem. You can't use pre-existing code for each component of the coupling

Partitioned approach

Each component of the problem is treated individually, resulting in as many algebraic problems as physical theories involved. The coupling conditions appear as source terms or boundary conditions, and you can use pre-existing codes. However, instabilities or/and inaccuracy may arise.

HPC context

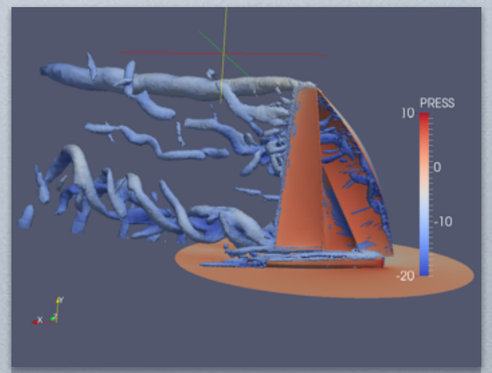
ALYA

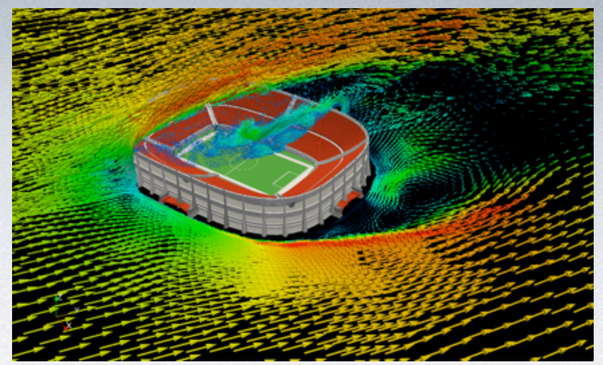
Multiphysics code

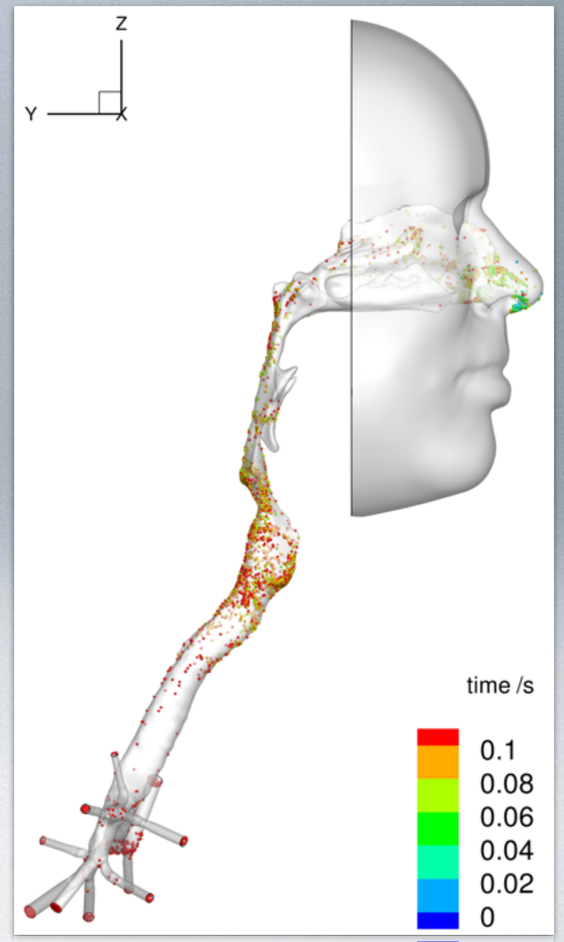
Finite Element Method (nodes)

Solids mechanics + Electrophysiology

Incompressible flow + ALE formulation







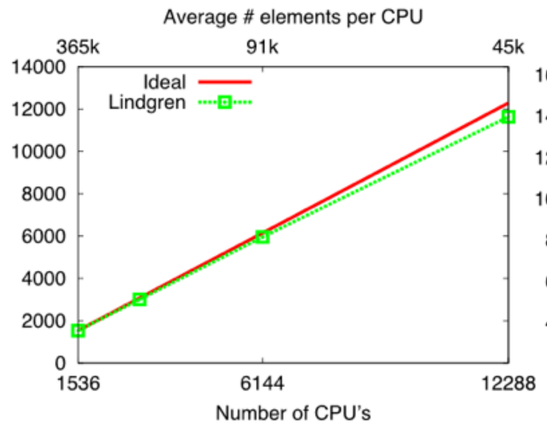
Challenges

=> Coupling strategy cannot be the bottleneck

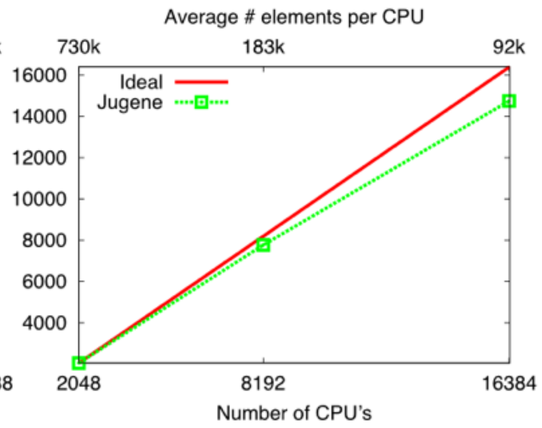
=> Multiphysics in exascale

HPC context (cont.)

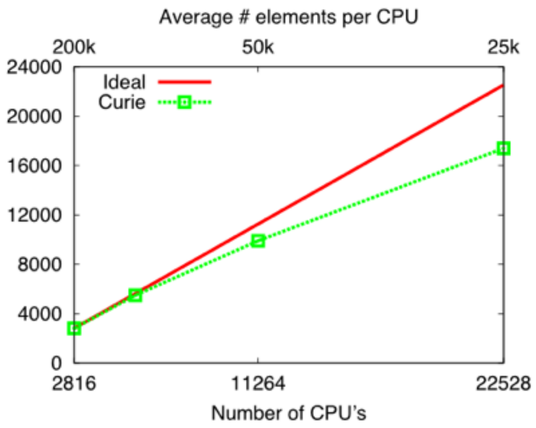
Lindgren - Cray XE6
Sweden



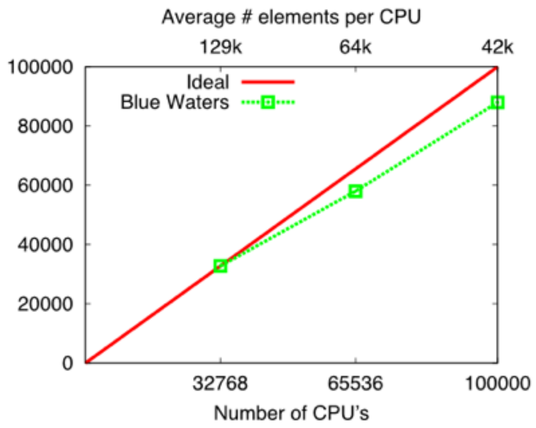
Jugene - Blue Gene/P
Germany



Curie - BullX
France

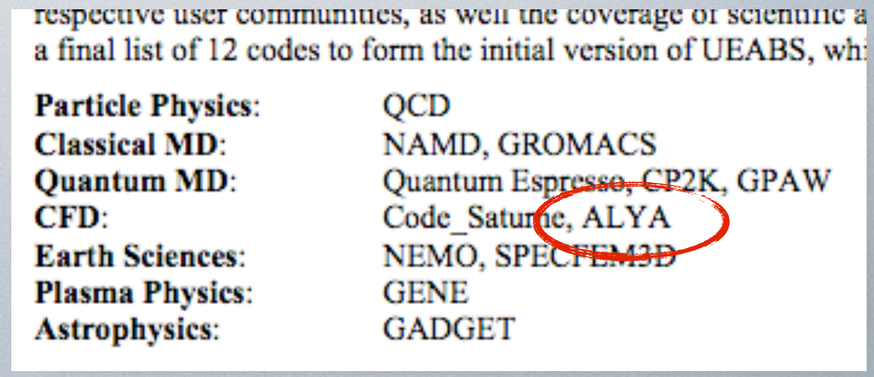


Blue Waters - Cray XE6
USA



HPC Simulation Tools: Alya

Alya is one of the two CFD codes of the PRACE benchmark suite


respective user communities, as well the coverage of scientific a
a final list of 12 codes to form the initial version of UEABS, wh

Particle Physics:	QCD
Classical MD:	NAMD, GROMACS
Quantum MD:	Quantum Espresso, CP2K, GPAW
CFD:	Code_Saturne, ALYA
Earth Sciences:	NEMO, SPECFEM3D
Plasma Physics:	GENE
Astrophysics:	GADGET

Lindgren (Sweden), Cray XE system at PDC, incompressible flow 12288 CPU's

(collaboration with Jing Gong from PDC)

Huygens, (The Netherlands), IBM power 6, incompressible flow, 2128 CPU's

Jugene BG (Germany): 16384 CPU's, incompressible flow (Prace project for Mesh multiplication) and, running first tests of FSI in collaboration with Paolo Crosetto (Julich)

Fermi BG (Italy): 16384 CPU's, incompressible flow + species transport + Lagrangian particles (Prace project for nose)

Curie Bullx (France): 22528 CPU's, incompressible flow (collaboration with Jing Gong - PDC)

Marenostrum: 5000 CPU's compressible flow, incompressible flow, thermal flow (scalability test)

BlueWaters: 100,000 CPU's combustion, elctro-mechanical model (collaboraiton with NCSA)

The ingredients

1. What

   The coupling variable to exchange

2. Where

   The coupling interface or volume

3. How

   The coupling algorithm

4. When

   The section of the code where the exchange will be performed

What

Unknown

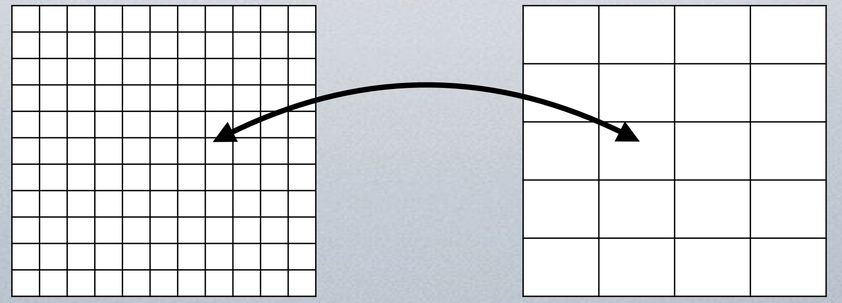
The unknowns of the equations are the coupling variables (velocity, temperature, pressure, displacement)

Residual

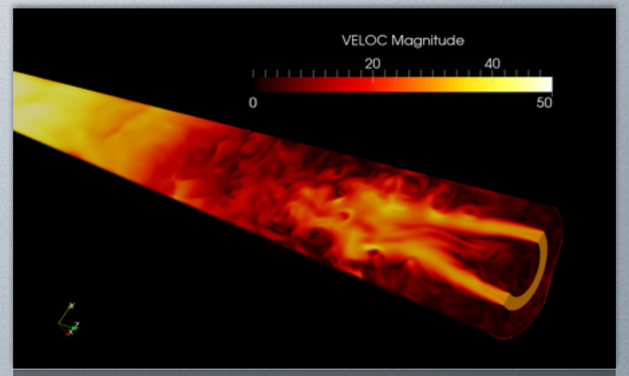
Residual coming from the unknowns in the physical equations are the coupling variables (heat flux, shear stress)

Where

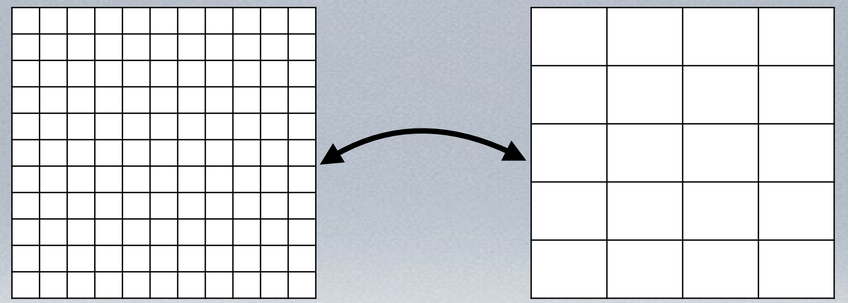
1. On Volume (Low Mach)



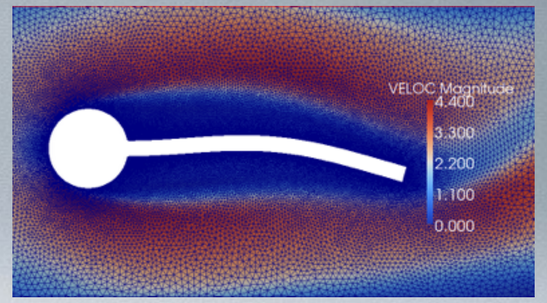
Navier-Stokes Temperature



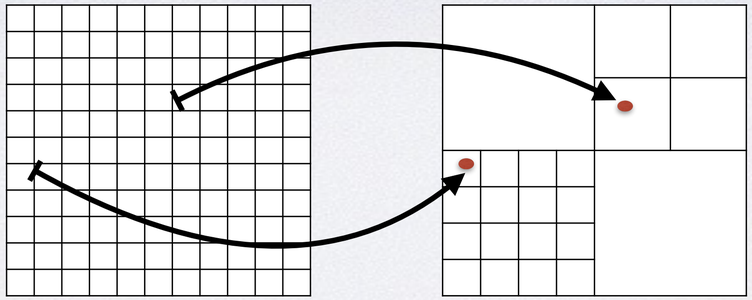
2. On Surface (FSI)



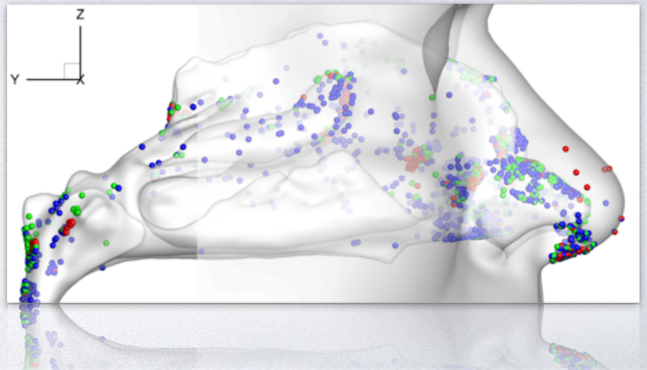
Navier-Stokes Solid mechanics



3. On points (Fluid/particle)

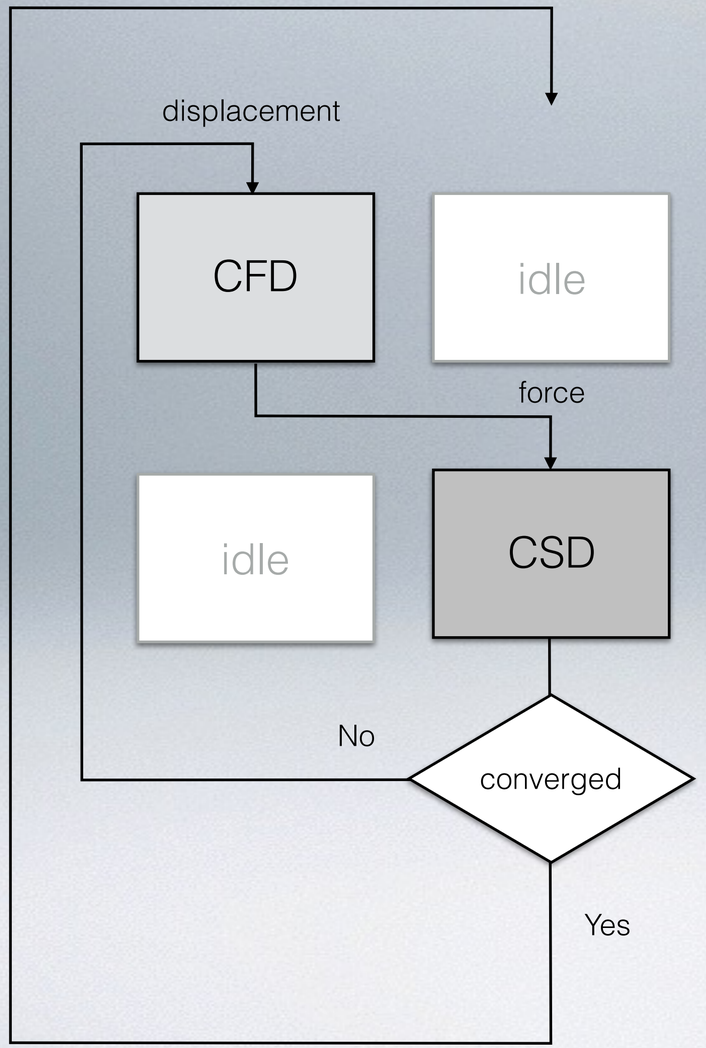


Navier-Stokes Particles

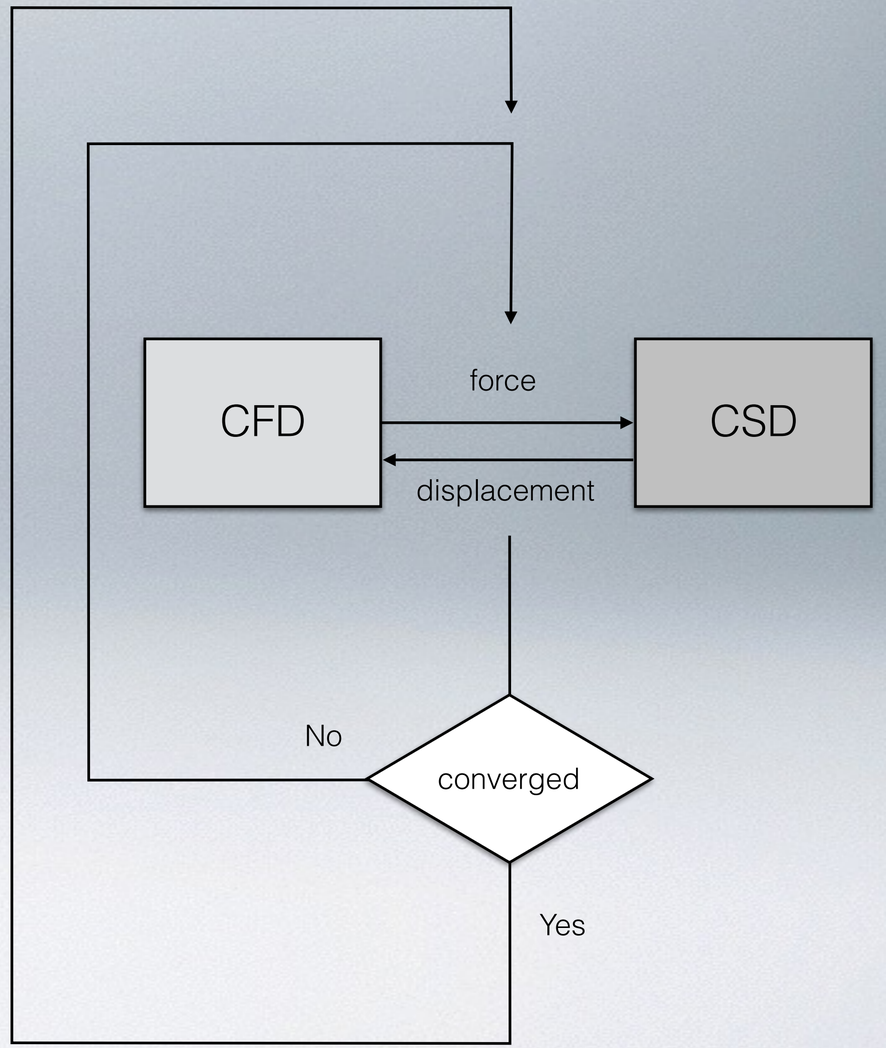


How

Gauss-Seidel approach



Jacobi approach



When

Pre-processing

Initialisation of coupling

Time loop

    Non-linear iterations loop

        Assemble matrix

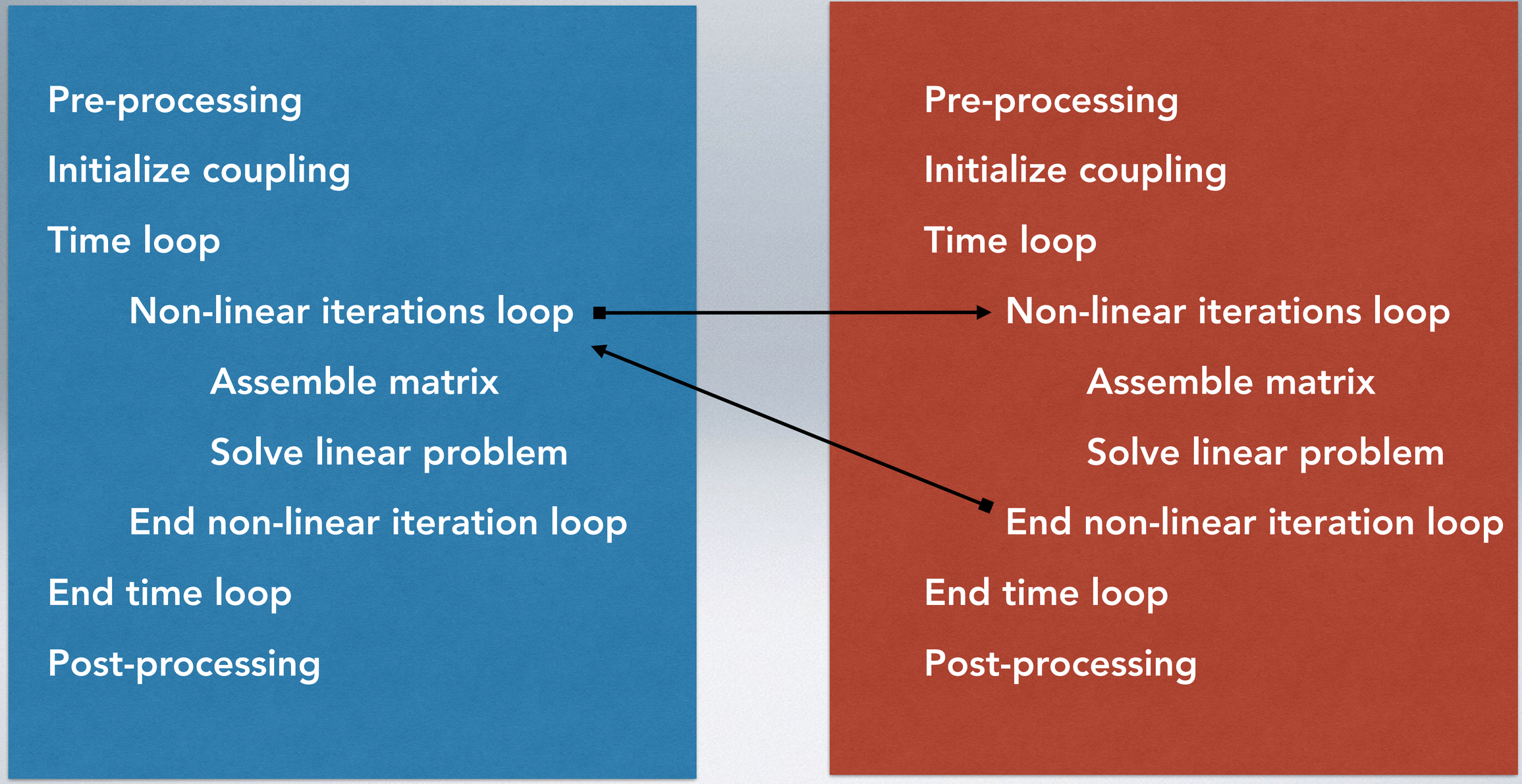
        Solve linear problem

    End non-linear iteration loop

End time loop

Post-processing

When



When

Pre-processing

Initialize coupling

Time loop

Non-linear iterations loop

Assemble matrix

Solve linear problem

End non-linear iteration loop

End time loop

Post-processing

Pre-processing

Initialize coupling

Time loop

Non-linear iterations loop

Assemble matrix

Solve linear problem

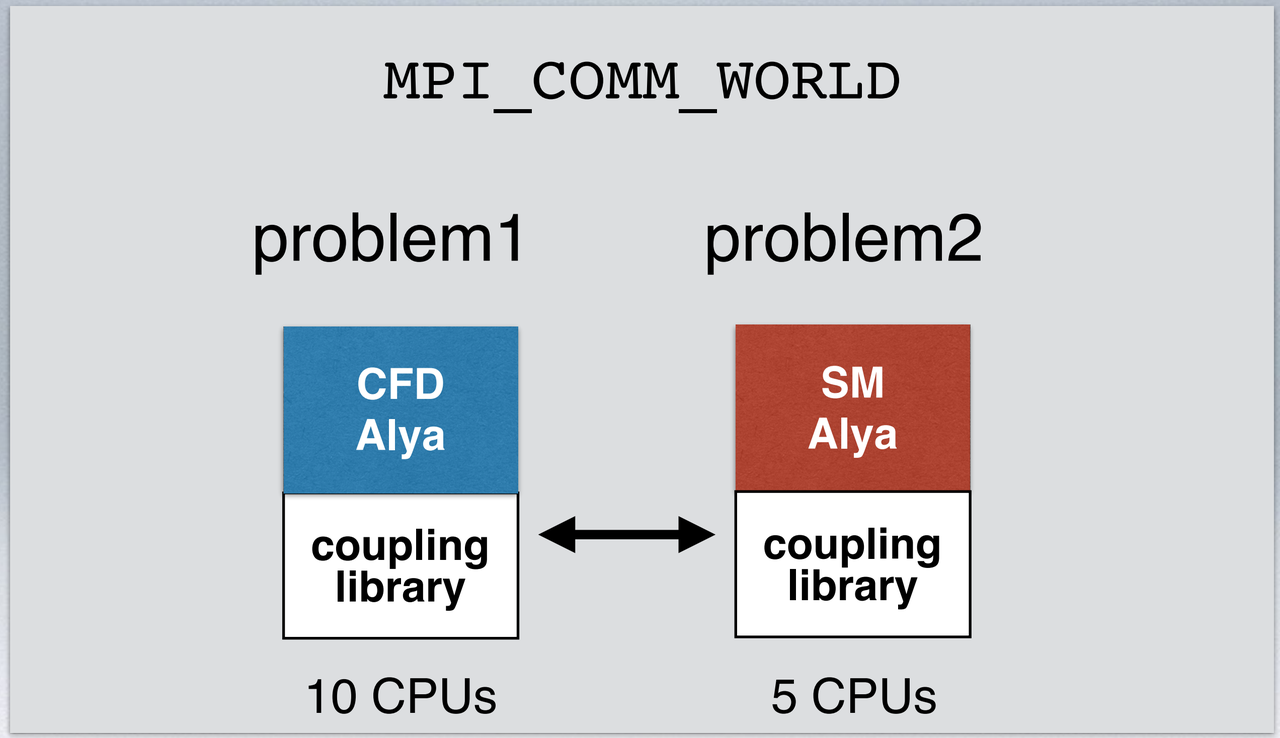
End non-linear iteration loop

End time loop

Post-processing

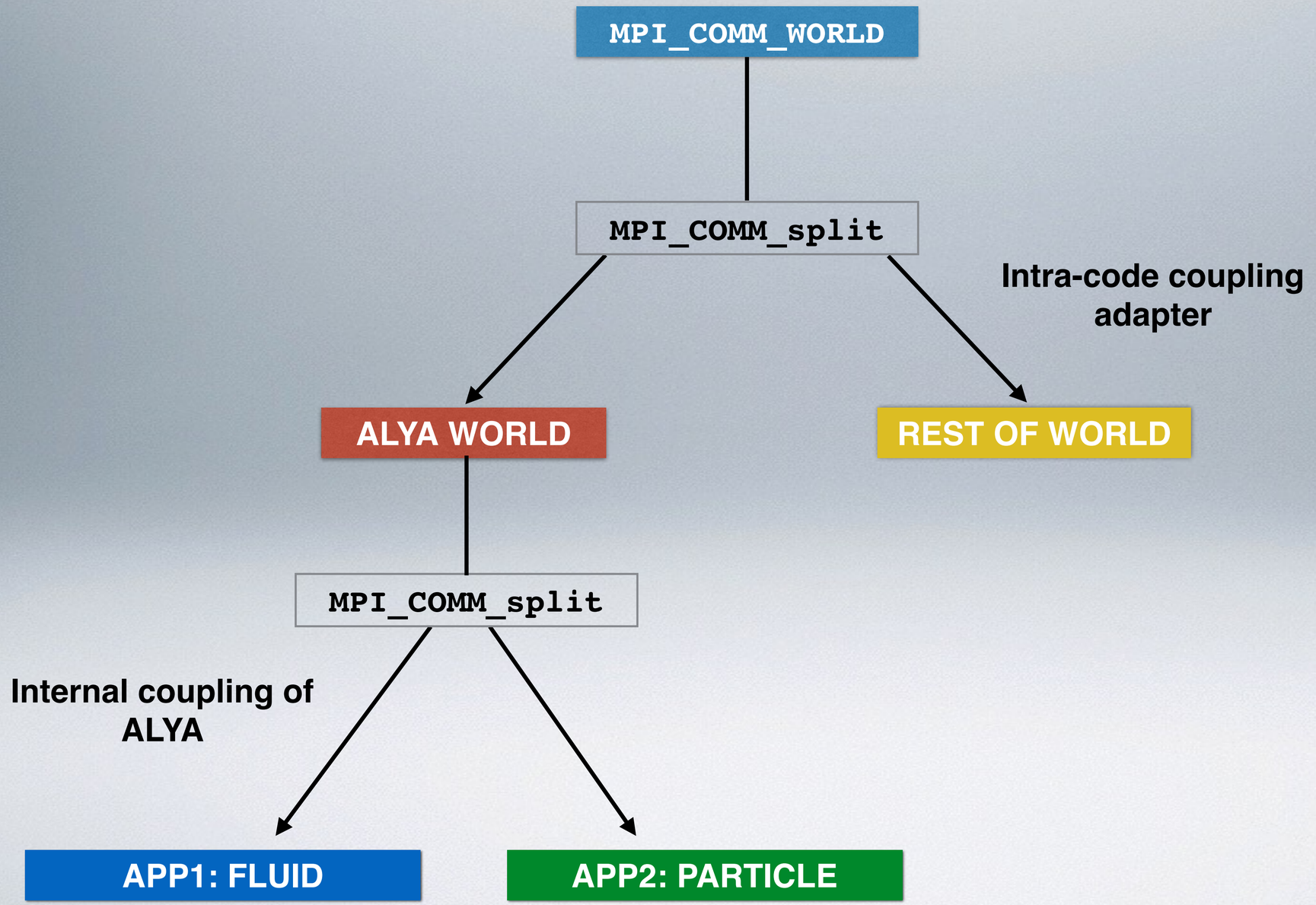
Parallel context

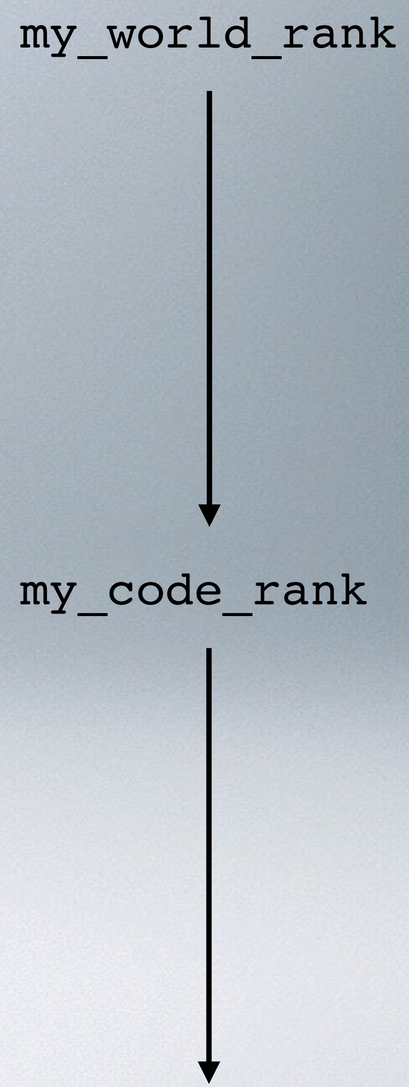
- ALYA coupling/PLE/PRECICE are library linked to the code
- Codes share the same `MPI_COMM_WORLD`
- Codes communicate through MPI messages




```
mpirun -n 10 Alya.x problem1 : -n 5 Alya.x problem2
```

Multi-code MPI environment

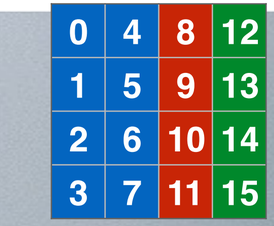




Multi-code MPI environment

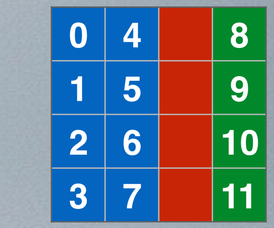
```
mpirun -n 8 Alya.x fluid : -n 4 Test.x : -n 4 Alya.x particle
```

```
call MPI_INIT()
call MPI_COMM_RANK(MPI_COMM_WORLD,my_world_rank,ierr)
call GETARG(0,app_name)
call GETARG(1,app_phys)
```



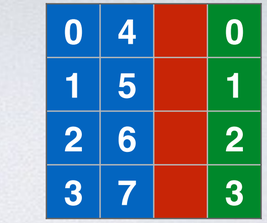
my_world_rank

```
if( app_name == 'Alya') then
  color = 1
else
  color = 0
end if
call MPI_COMM_SPLIT(MPI_COMM_WORLD,color,my_world_rank,COMM_CODE,ierr)
call MPI_COMM_RANK(COMM_CODE,my_code_rank,ierr)
```



my_code_rank

```
if( app_name == 'Alya') then
  if( app_phys == 'fluid') then
    color = 1
  else if( app_phys == 'particle')  then
    color = 0
  end if
  call MPI_COMM_SPLIT(COMM_CODE,color,my_world_rank,COMM_PHYS,ierr)
  call MPI_COMM_RANK(COMM_PHYS,my_phys_rank,ierr)
end if
```



my_phys_rank

Initialize coupling

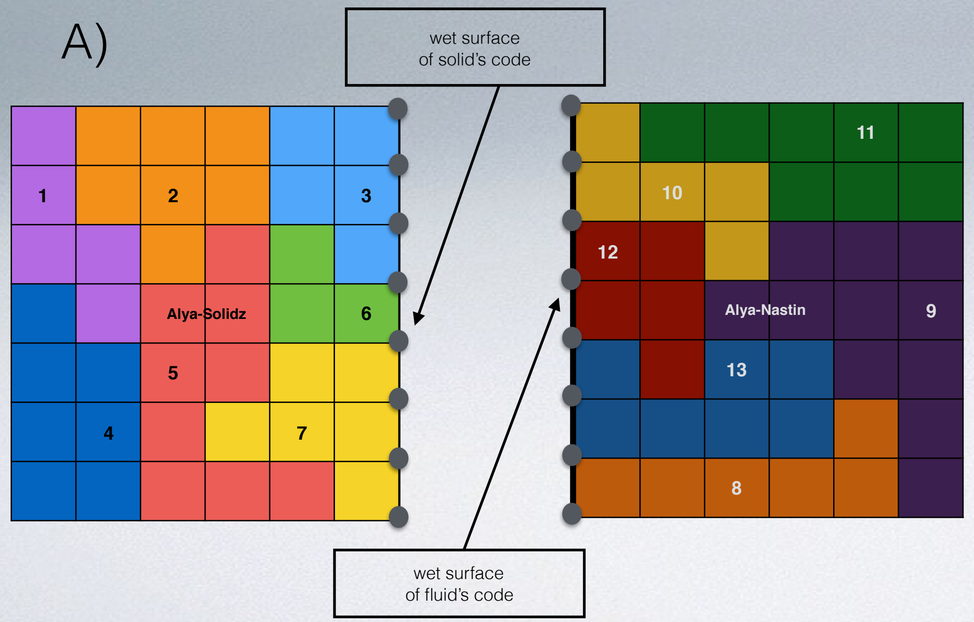
A. Wet entity (volume, surface, point) => define a set of target points (nodes, Gauss points)

B. Identify candidate host CPUs for target point

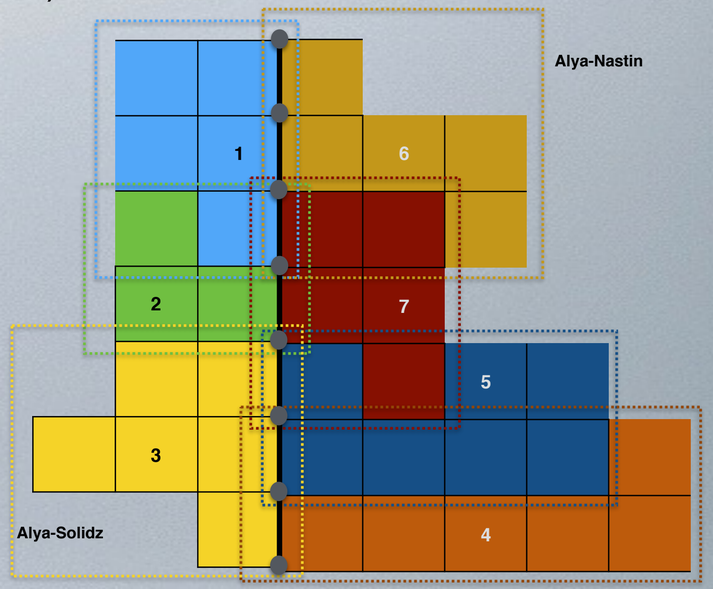
- Send my target points to candidates
- Receive answer from candidates
- Decide who's the owner CPUs

C. Set communication strategy and arrays

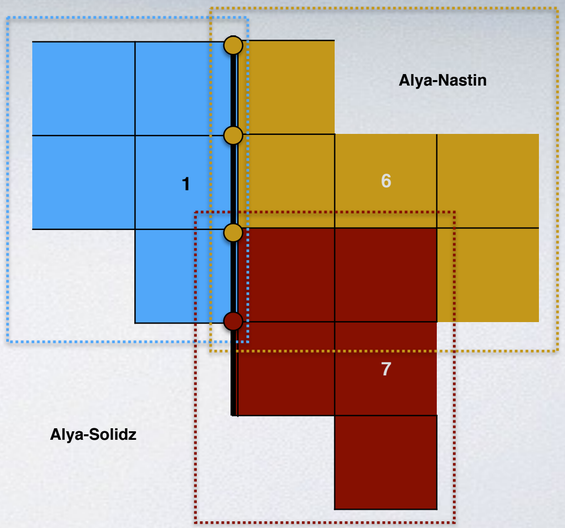
A)



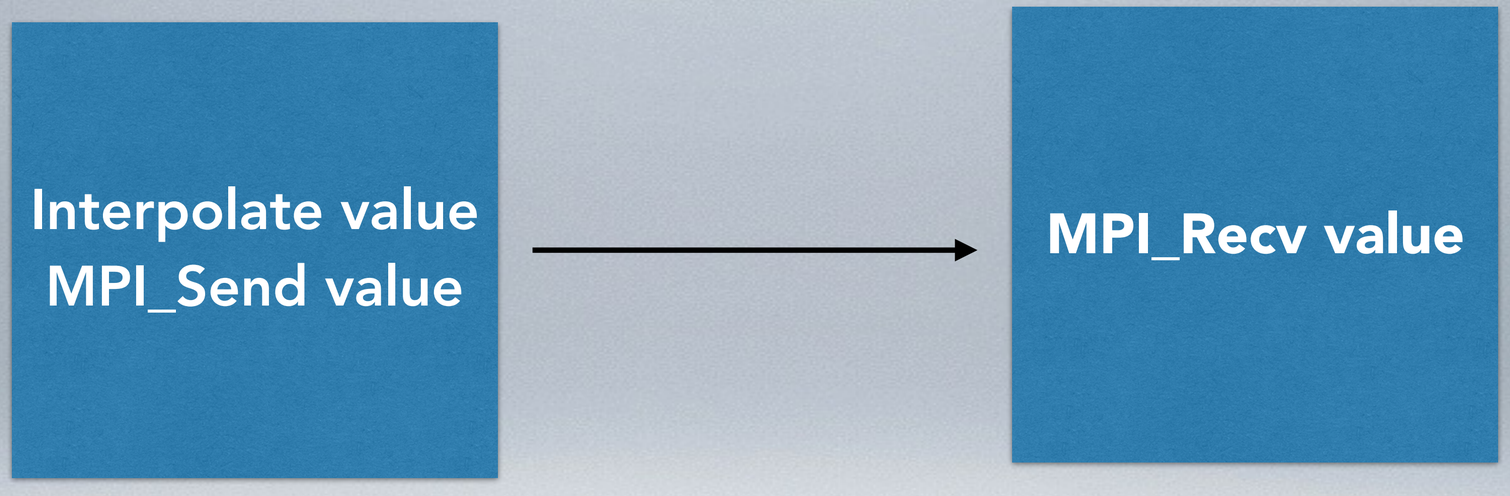
B)



C)



Send - Receive



Compute and send

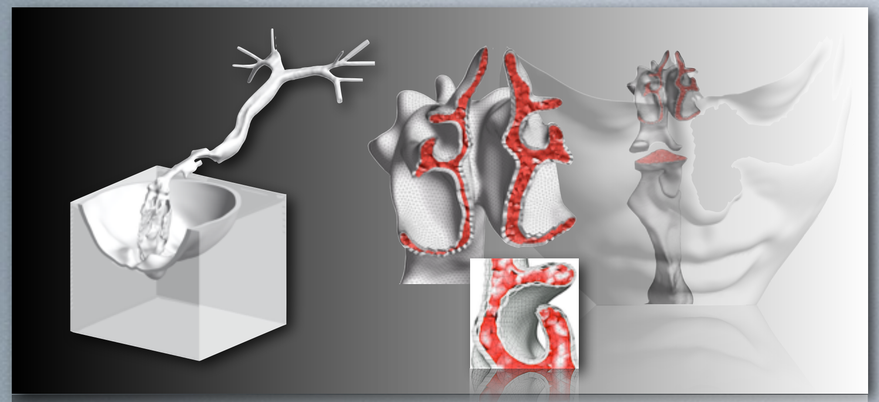
Receive and assemble

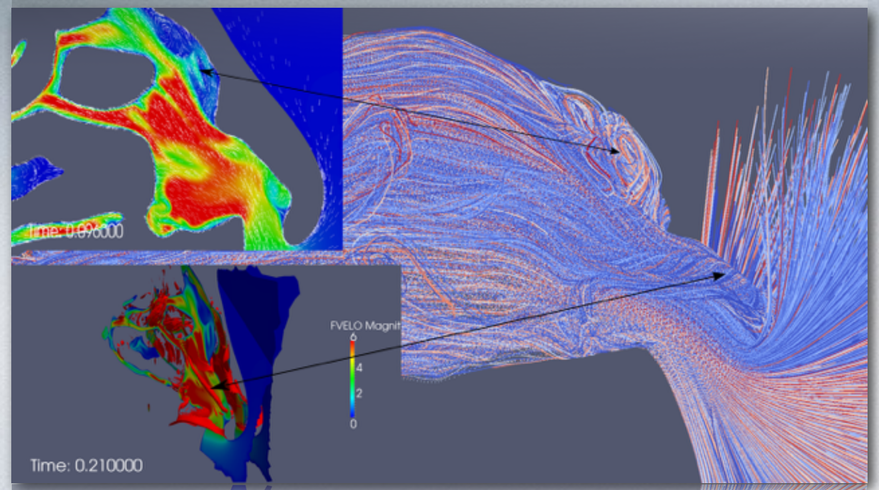
Parallel & Asynchronous Coupling of Fluid and Particle Solver

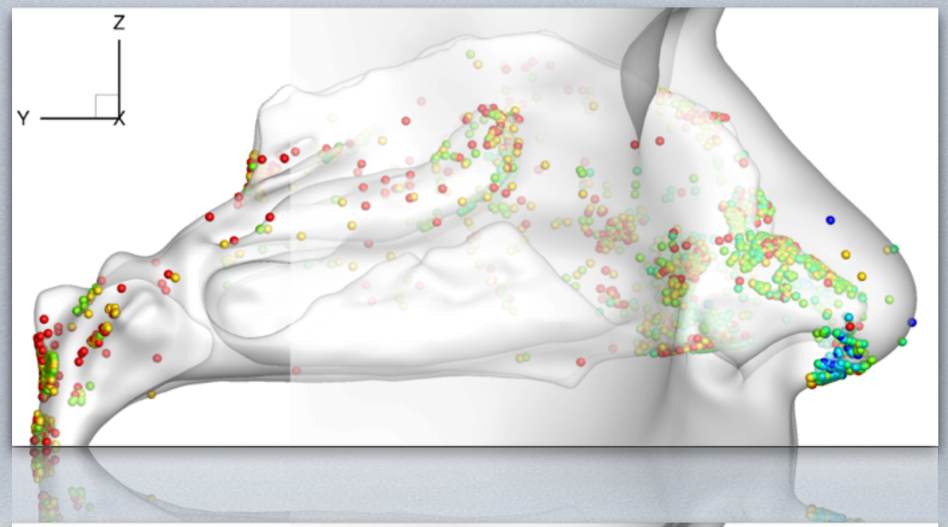
Target application: particle deposition in respiratory system

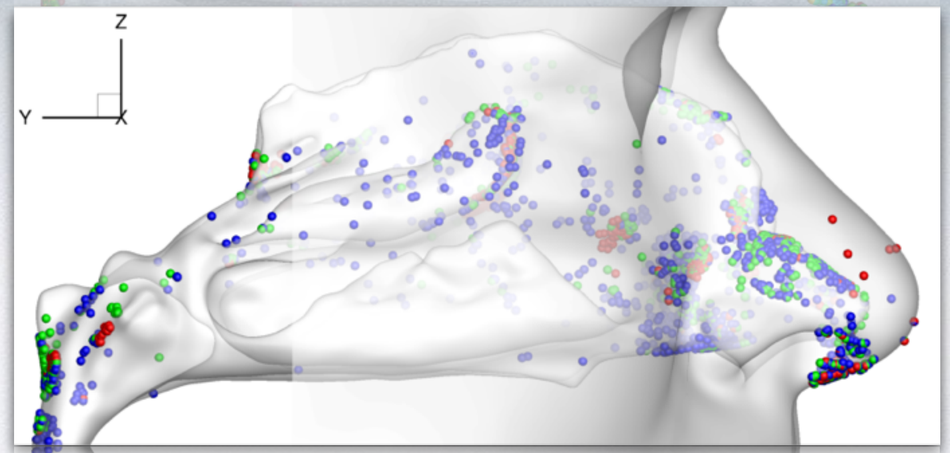
Drug delivery, spray design, toxicology…

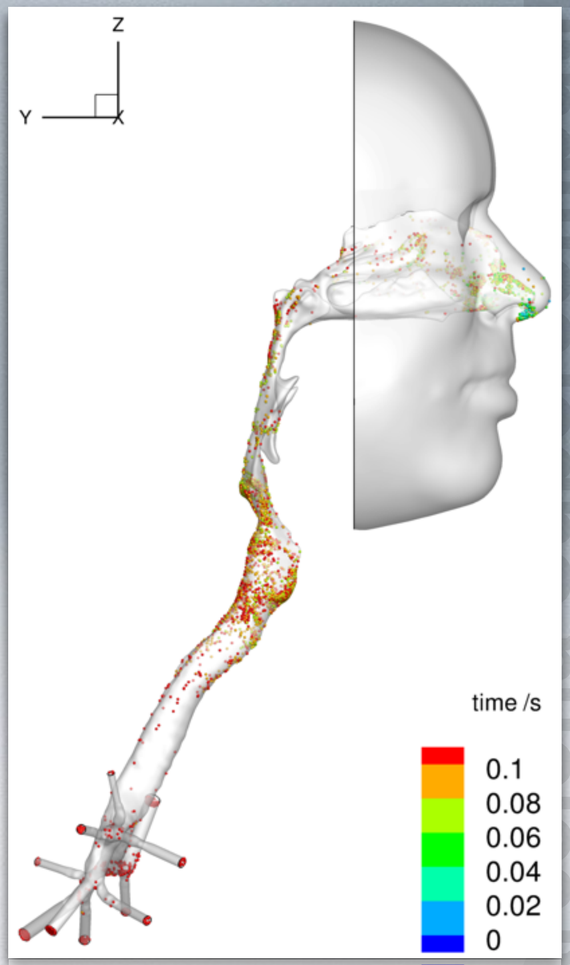
DNS, 0.5B elements, 10^5 time steps, 10^6 particles





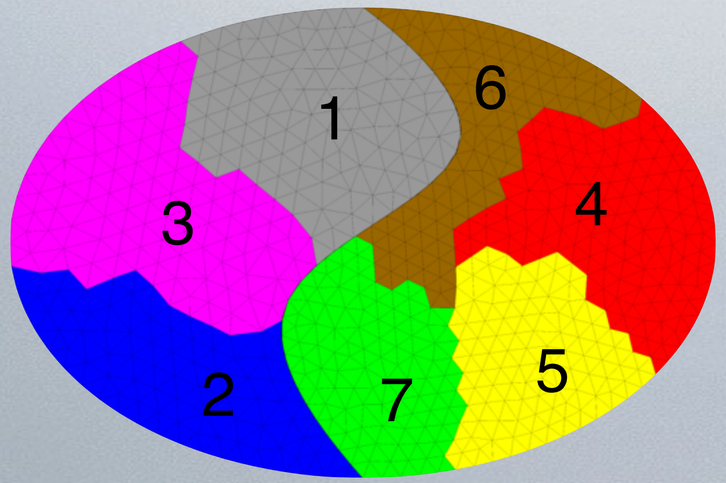






Shortcomings of using same instance of a code

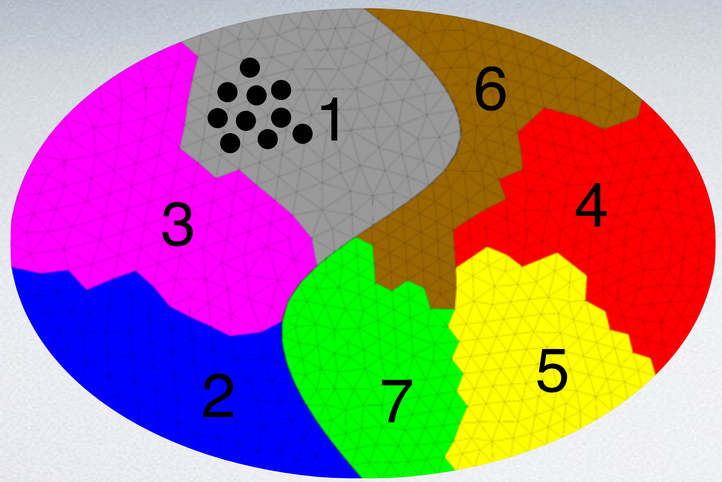
Solve fluid



Mesh partition is adapted to fluid
Good load balance
Good scalability

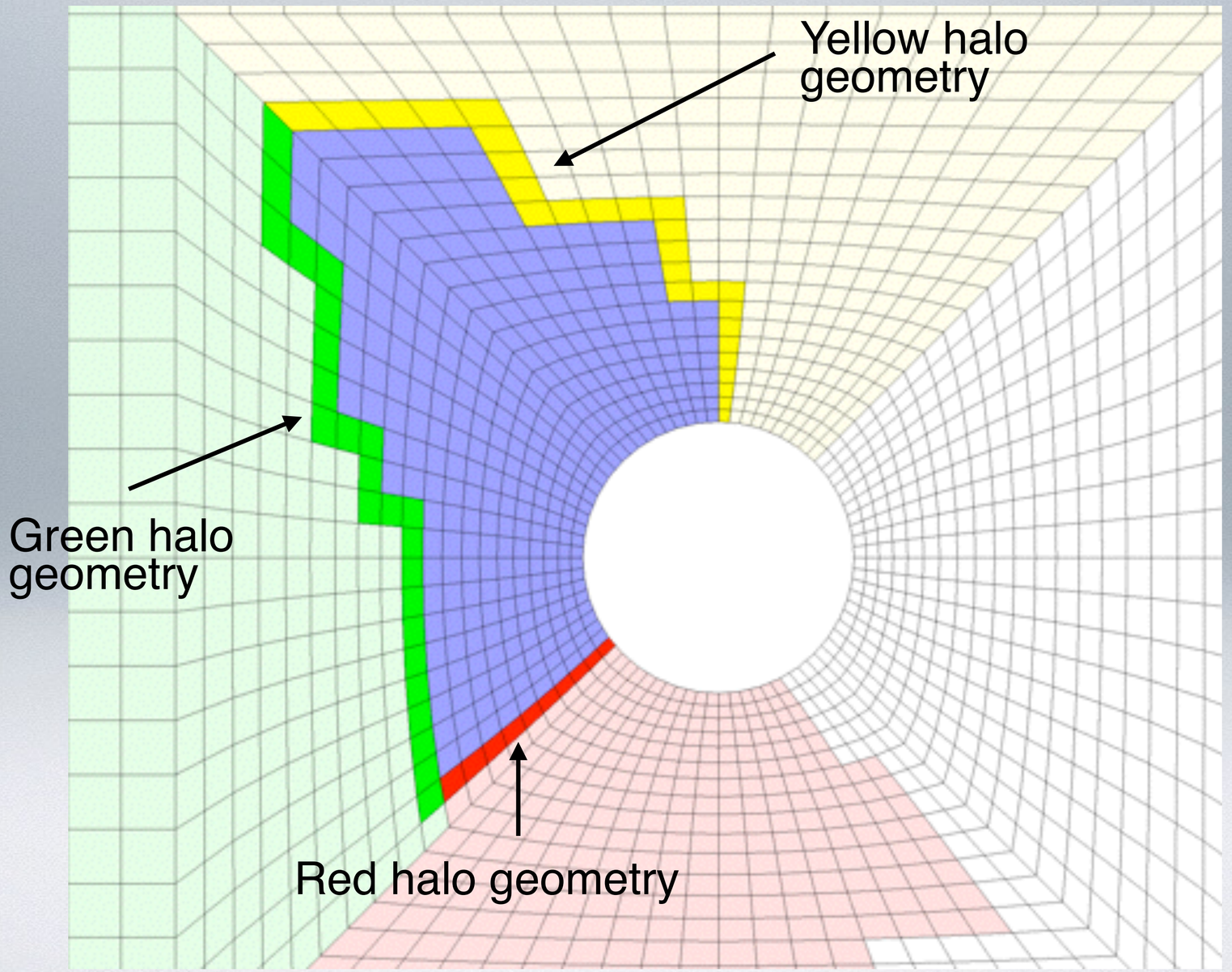
Inject particles

Transport particles



Particles may be concentrated in few CPUs
Bad load balance
Bad scalability

Parallelization

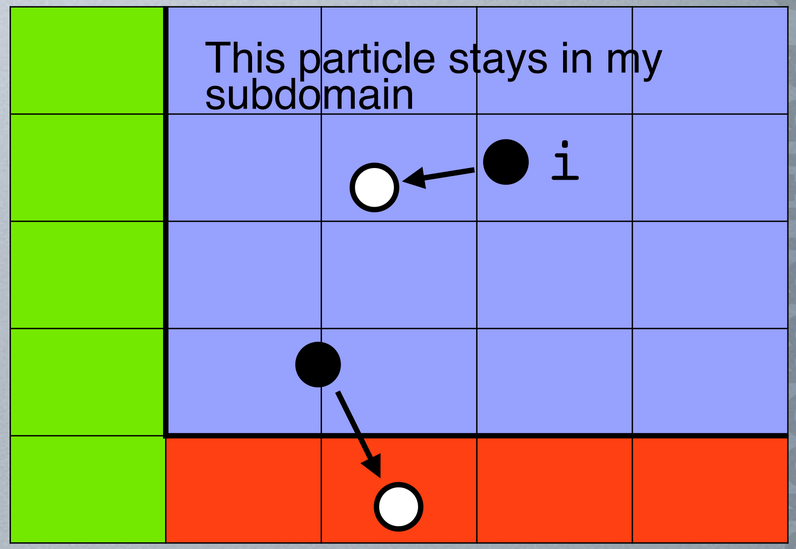


Parallelization

```
do while nsend /= 0
   nsend = 0
   !$OMP PARALLEL
   !$OMP DO
   do i = 1,n
     do while t /= t_f
       Update x_i^{n+1}
       if i is in halo element then
         nsend = nsend + 1
       end if
     end do
   end do
   MPI_AllReduce(nsend)
   MPI_Send particles in halo elements to neighbors
end do
```

loop over particles i

communication loop



This particle goes to a halo element and will be sent to my green neighbor in next communication step

Coupling strategy

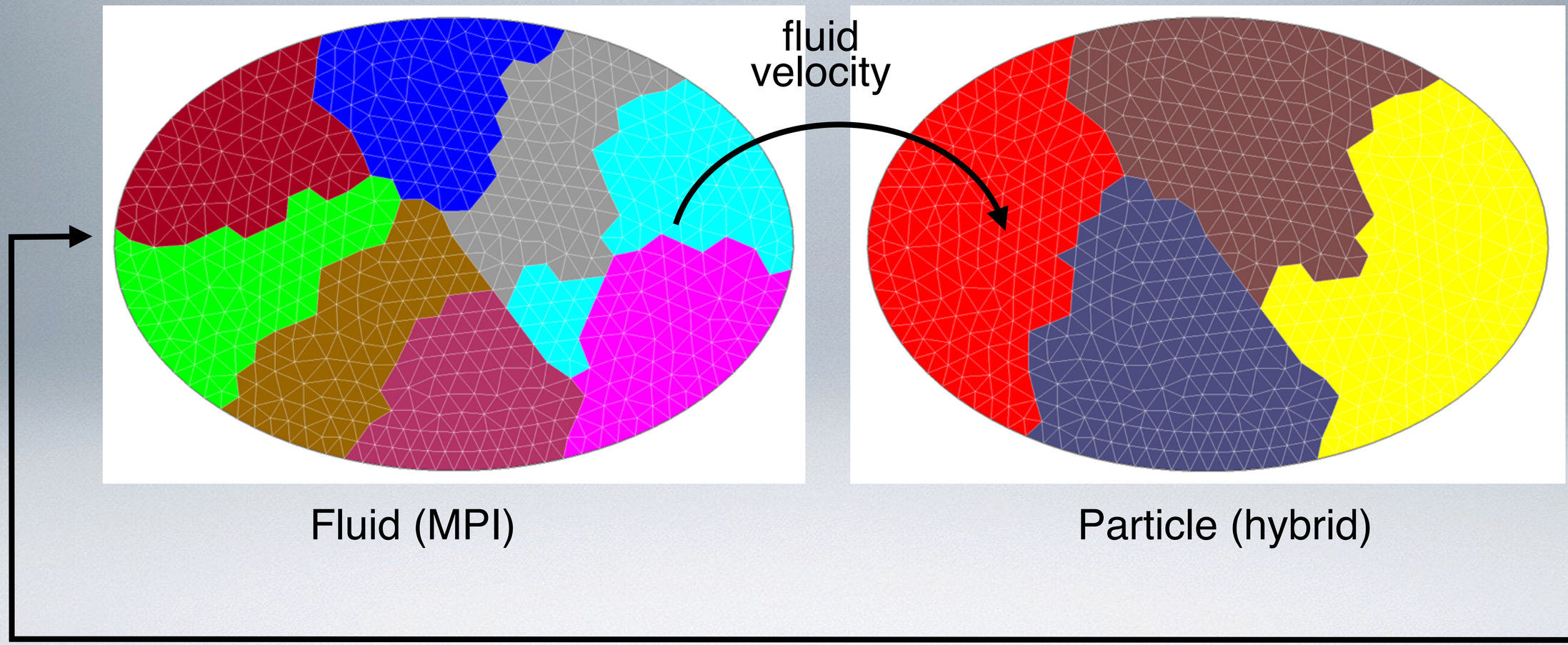
Problems

- Particles may be concentrated in very few MPI tasks
- One cannot predict where particles are going
- Important load imbalance and synchronization

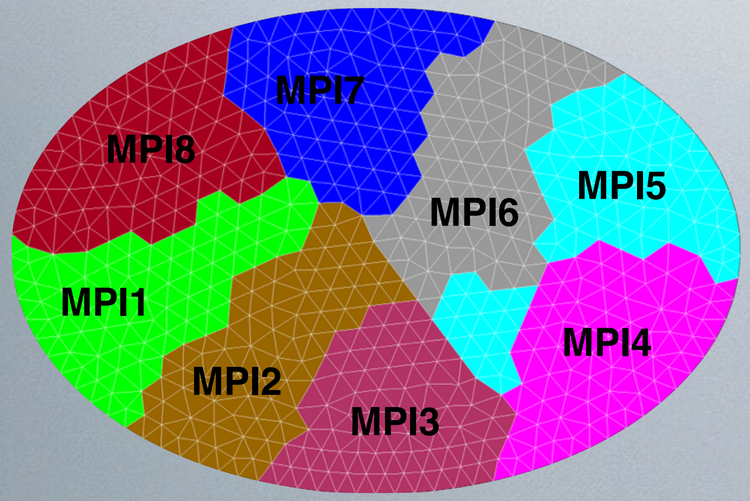
Solutions

- Multi-code strategy to achieve coarse grain ***asynchronism***
- ***Dynamic load balancing*** to load balance particles at runtime

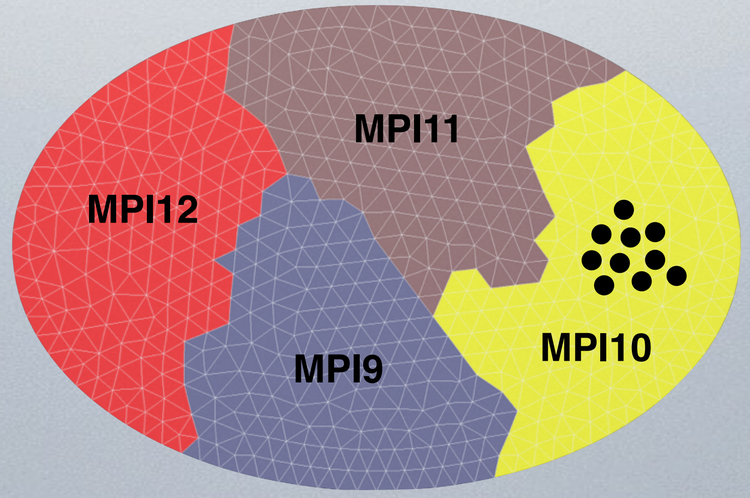
Coupling strategy



Need for dynamic load balance

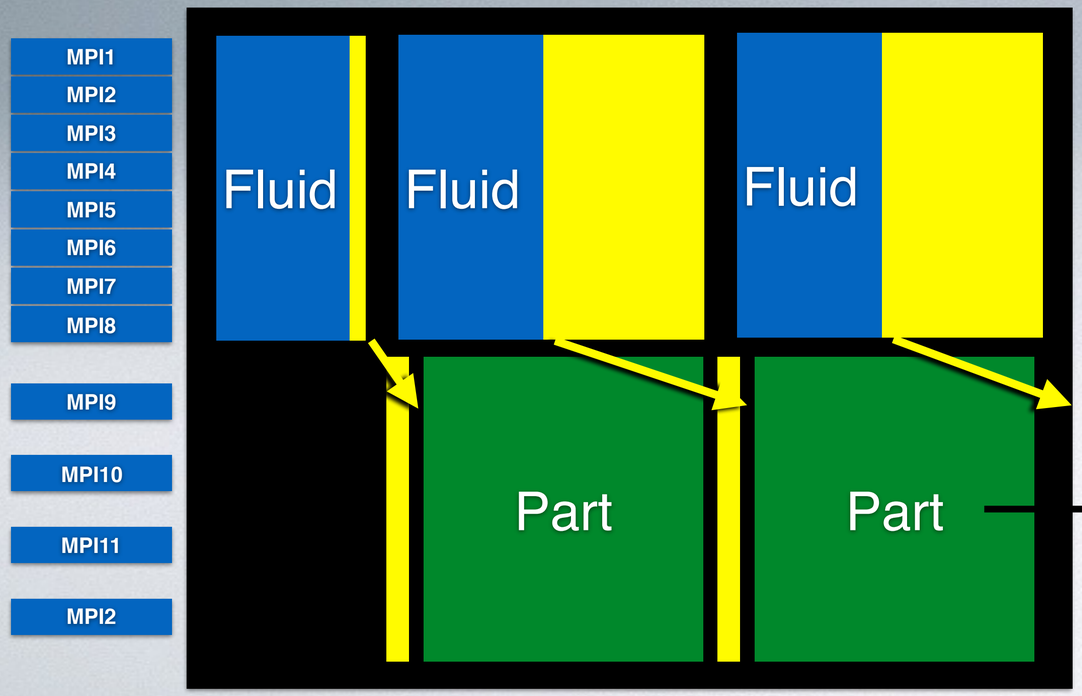


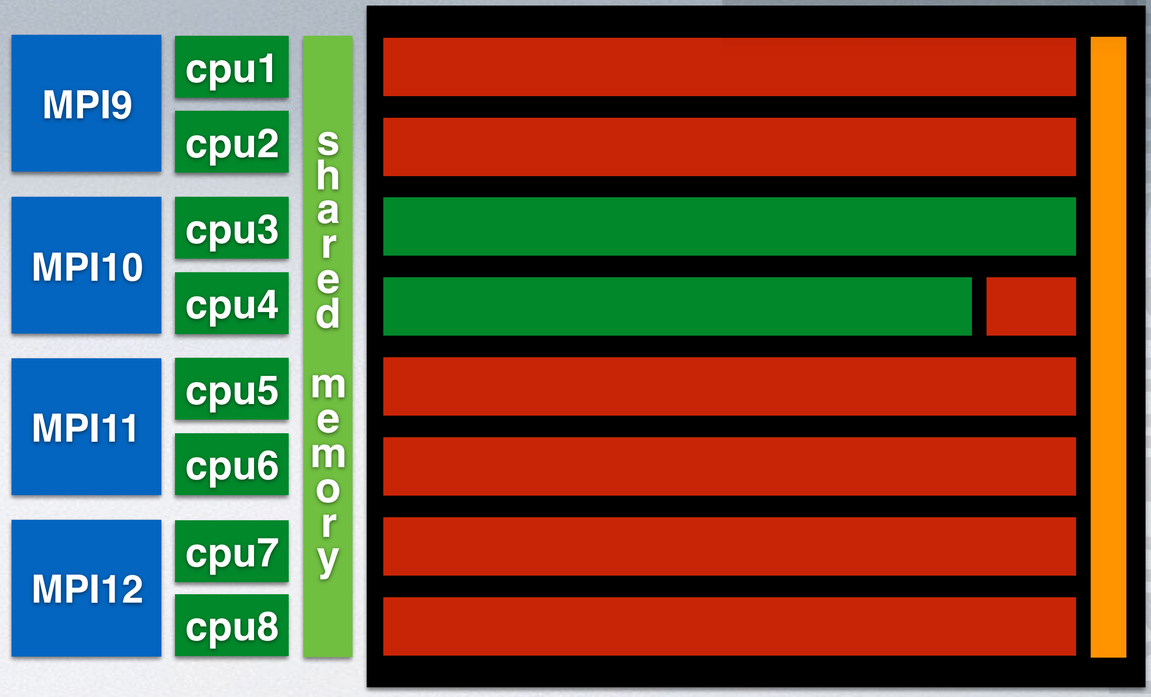
Fluid



Particle

point-to-point comm.
global comm.
working CPU (fluid)
working CPU (particle)
idle CPU





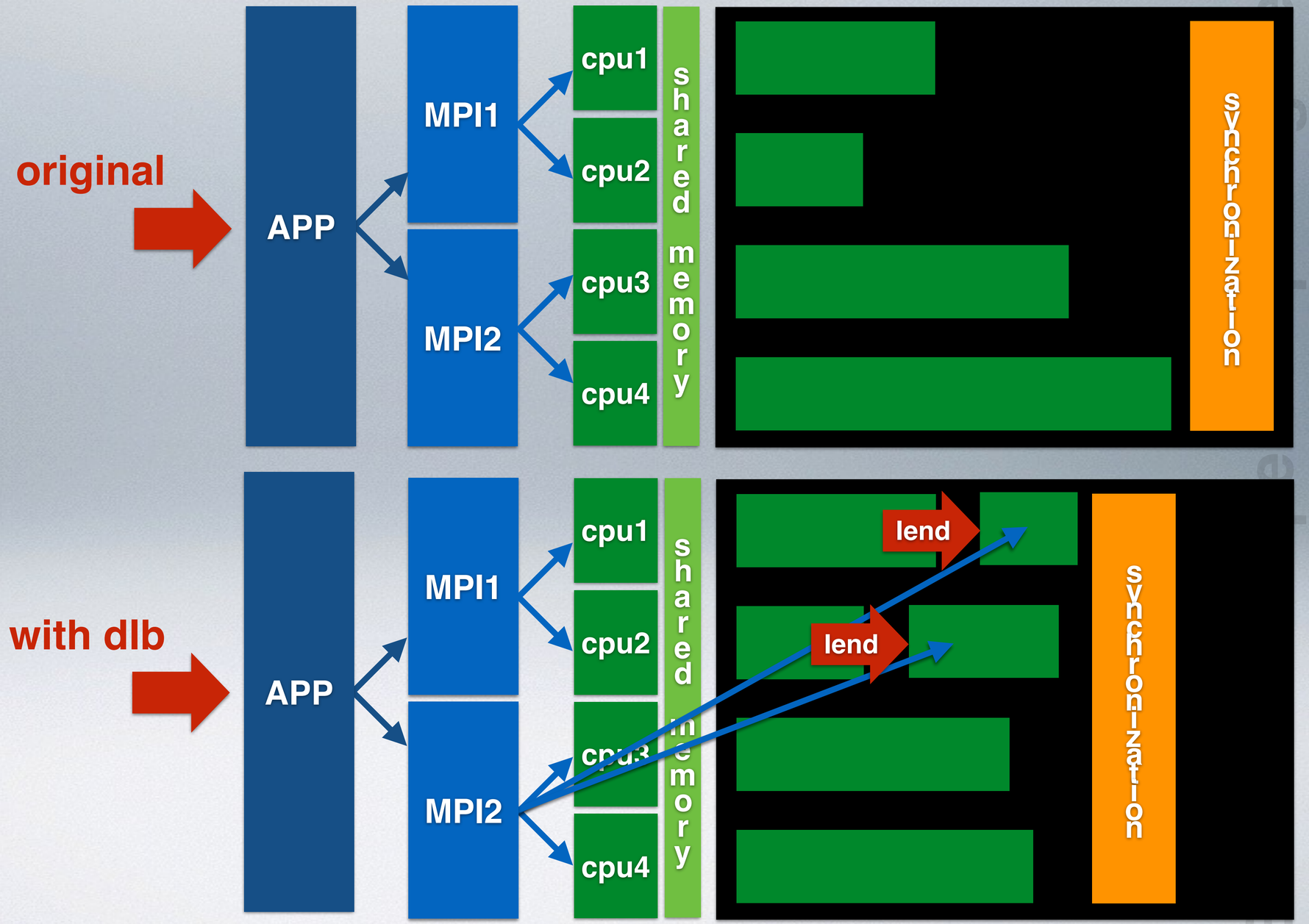
What is dlb

- dlb is a runtime library to enable dynamic load balancing
- dlb is developed at BSC-CNS
- dlb has a nice logo :o
- dlb understands OpenMP pragmas
- dlb enable to use resources from other MPI task whenever available
- dlb improves the performance of hybrid (MPI/OpenMP) codes without introducing modifications to original code

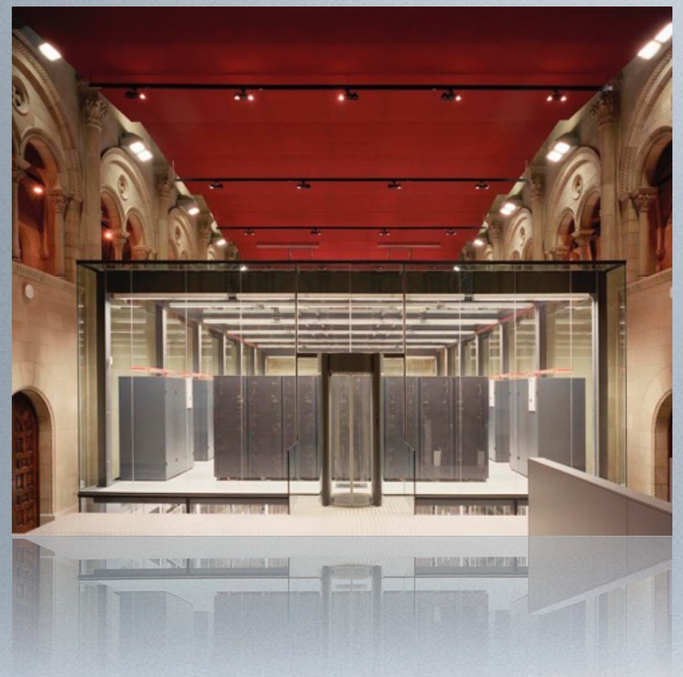
How dlb works

- Runtime library:
  - no previous modifications needed
  - dynamic decisions at execution time

- Lend resources when not using them
  - to another process in the same node



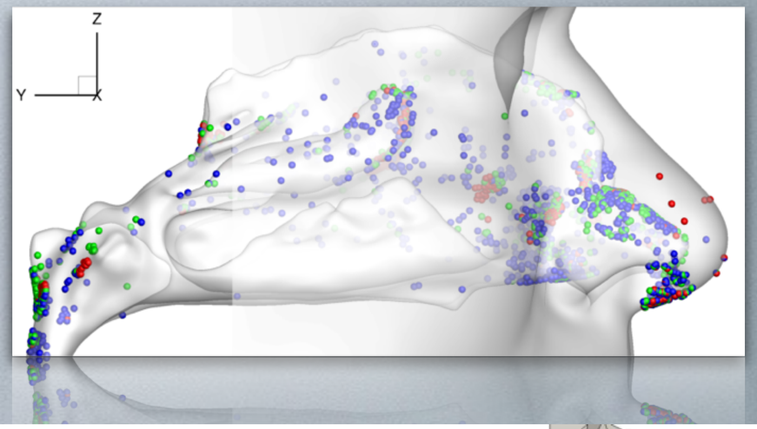
Configuration

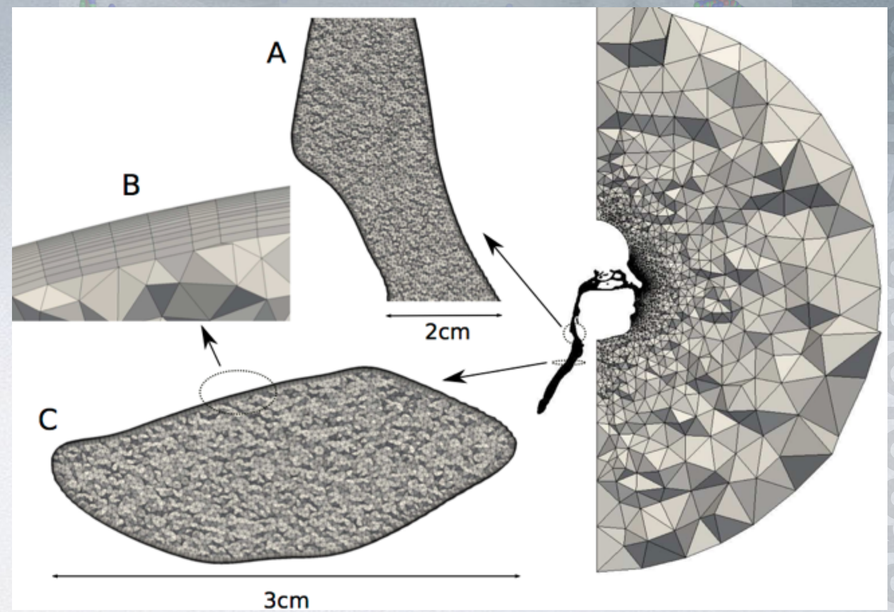


Peak Performance of 1,1 Petaflops
100.8 TB of main memory
3,056 compute nodes
2x Intel SandyBridge - 8-core at 2.6 GH

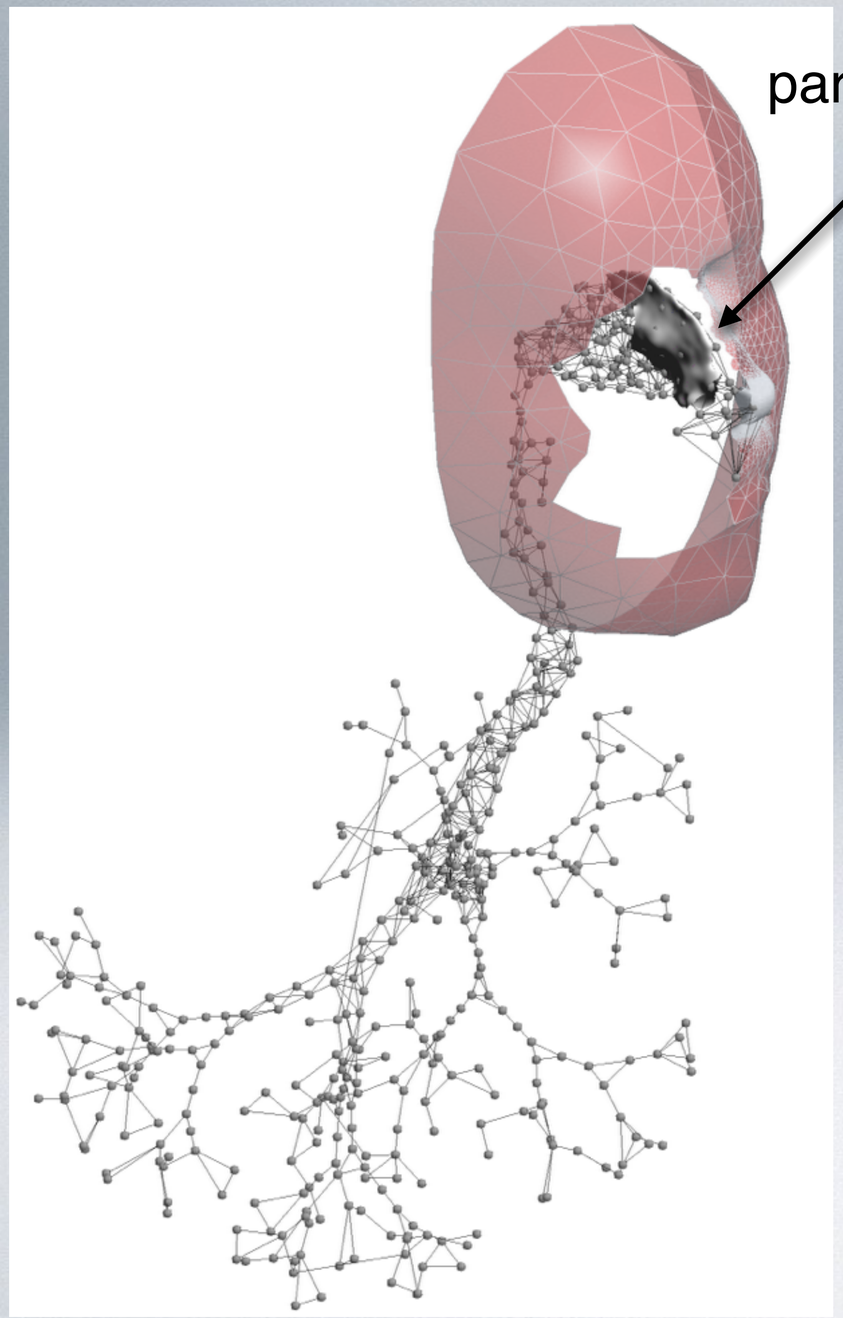
	MPI tasks	Threads	Nodes	#elements/ MPI task
Fluid	200	1	12,5	88k
Particle	72	2	9	246k

- 17M elements (tetras, prisms, pyramids)
- 2M particles

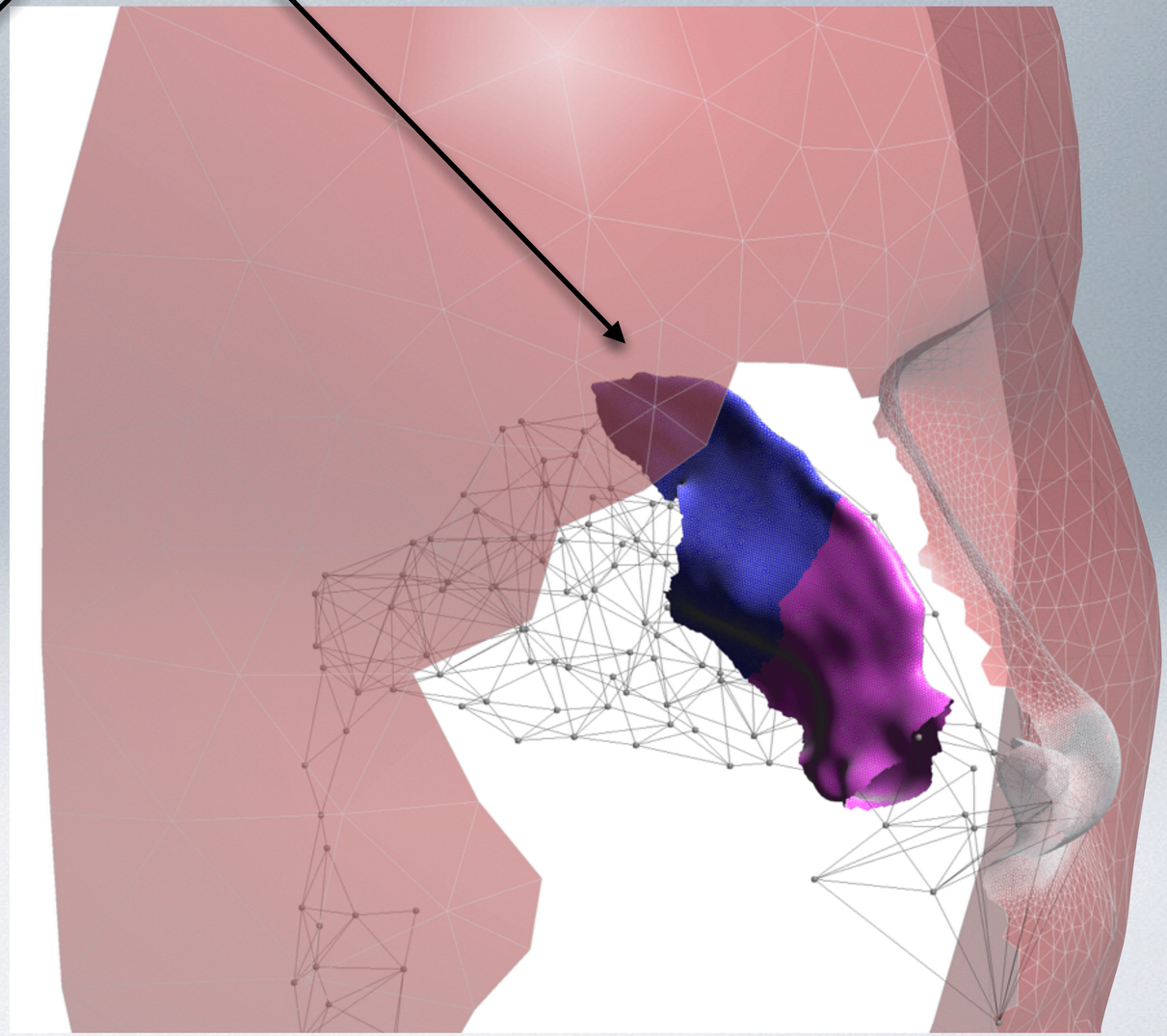




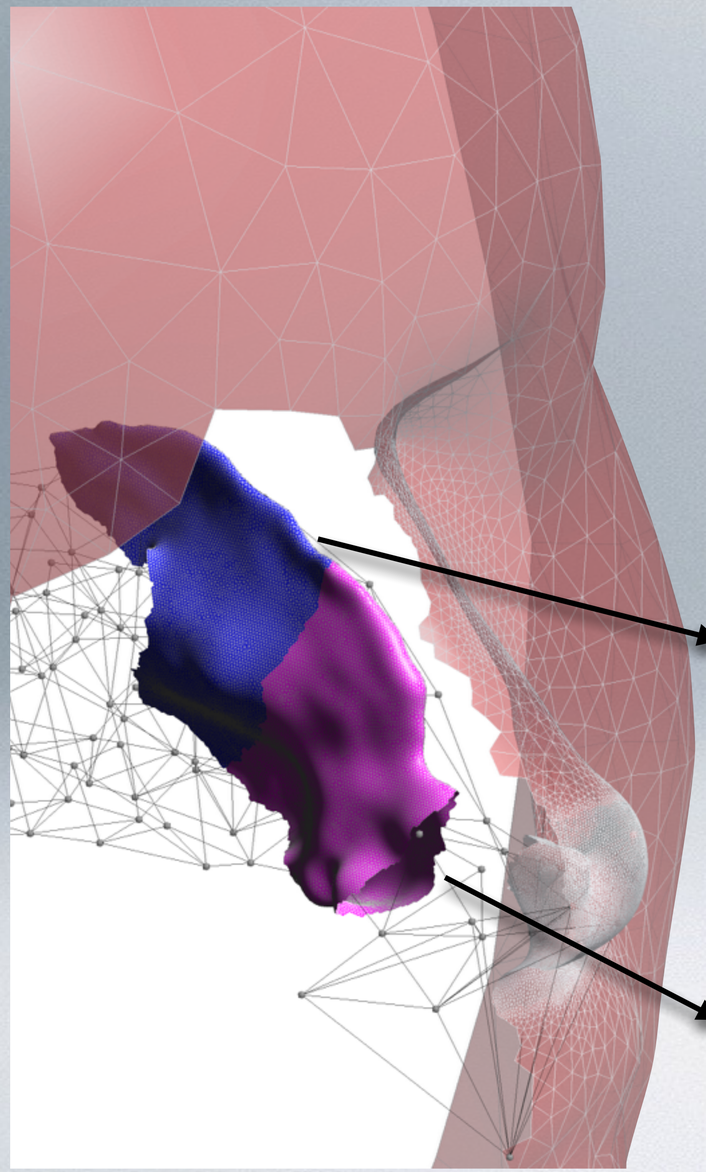
Particle subdomains



particles are concentrated in two subdomains



Dynamic load balancing

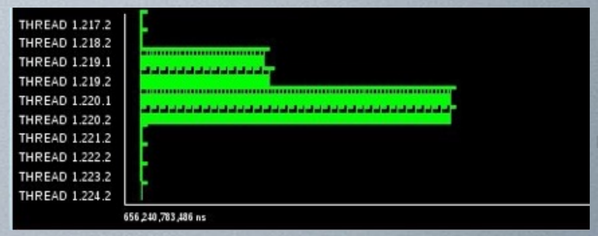


original: OMP_NUM_THREADS=2

only these MPI tasks have particles

MPI217 idle	idle	MPI218 idle	idle
MPI219 cpu1	MPI219 cpu2	MPI220 cpu1	MPI220 cpu2
MPI221 idle	idle	MPI222 idle	idle
MPI223 idle	idle	MPI224 idle	idle

1 Marenostrum node



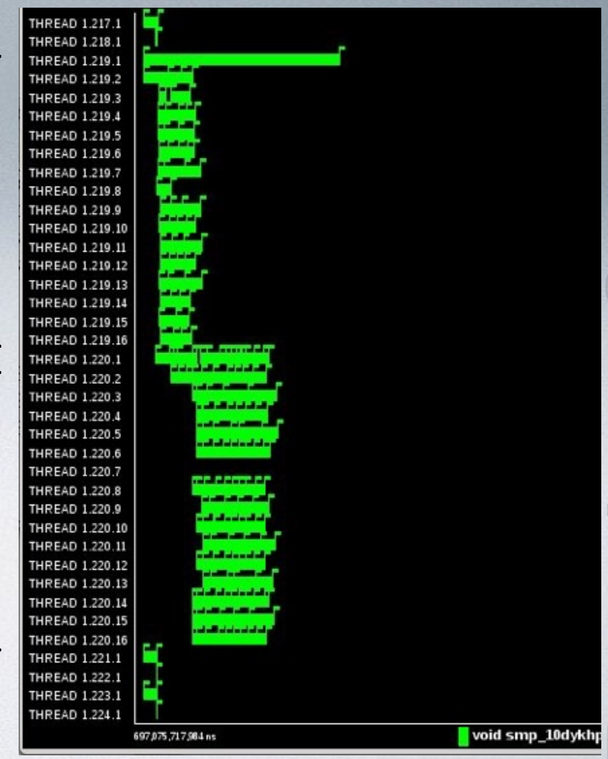
with dlb: OMP_NUM_THREADS=16

cpu's lent to MPI219

MPI219 cpu2	MPI219 cpu3	MPI219 cpu4	MPI219 cpu5
MPI219 cpu1	MPI219 cpu6	MPI219 cpu7	MPI219 cpu8
MPI219 cpu9	MPI219 cpu10	MPI219 cpu11	MPI219 cpu12
MPI219 cpu13	MPI219 cpu14	MPI219 cpu15	MPI219 cpu16

cpu's lent to MPI220

MPI220 cpu2	MPI220 cpu3	MPI220 cpu4	MPI220 cpu5
MPI220 cpu6	MPI220 cpu7	MPI220 cpu1	MPI220 cpu8
MPI220 cpu9	MPI220 cpu10	MPI220 cpu11	MPI220 cpu12
MPI220 cpu13	MPI220 cpu14	MPI220 cpu15	MPI220 cpu16



Visualization of airflow through the human respiratory system

Computational mesh: 44 million elements
80 thousand Lagrangian-Eulerian particles

Barcelona Supercomputing Center
Scientific Visualization Team

Parallel & Asynchronous Coupling of Fluid Structure Interaction problems

FSI problem

- Different physical descriptions
- Different mesh sizes
- Different numerical treatments
- Different time scales
- Physical and numerical instabilities

Fluid dynamics equations

$$\nabla \cdot \mathbf{u} = 0$$

$$\rho_f \frac{\partial \mathbf{u}}{\partial t} + \rho_f (\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla \cdot [2\mu\varepsilon(\mathbf{u})] + \nabla p = \rho \mathbf{f}$$

$\mathbf{u}$ is the velocity field in the fluid domain

ρ_f is the fluid's density

μ is the fluid's viscosity

ε is the velocity strain rate tensor

p is the pressure field in the fluid domain

Solid dynamics equations

$$\rho_s \frac{\partial \mathbf{d}_s}{\partial t} = \nabla \cdot \mathbf{P} + \mathbf{b}$$

$\mathbf{d}_s$ is the displacement field in the solid domain

ρ_s is the solid's density

$\mathbf{P}$ is the first Piola-Kirchhoff stress tensor

$\mathbf{b}$ is the body force

Coupling conditions: at the interface

$$\mathbf{d_f} = \mathbf{d_s}$$

$$\mathbf{t_f} = -\mathbf{t_s}$$

$\mathbf{d_f}$ is the displacement of the interface in the fluid domain

$\mathbf{d_s}$ is the displacement of the interface in the solid domain

$\mathbf{t_f}$ is the traction on the interface in the fluid domain

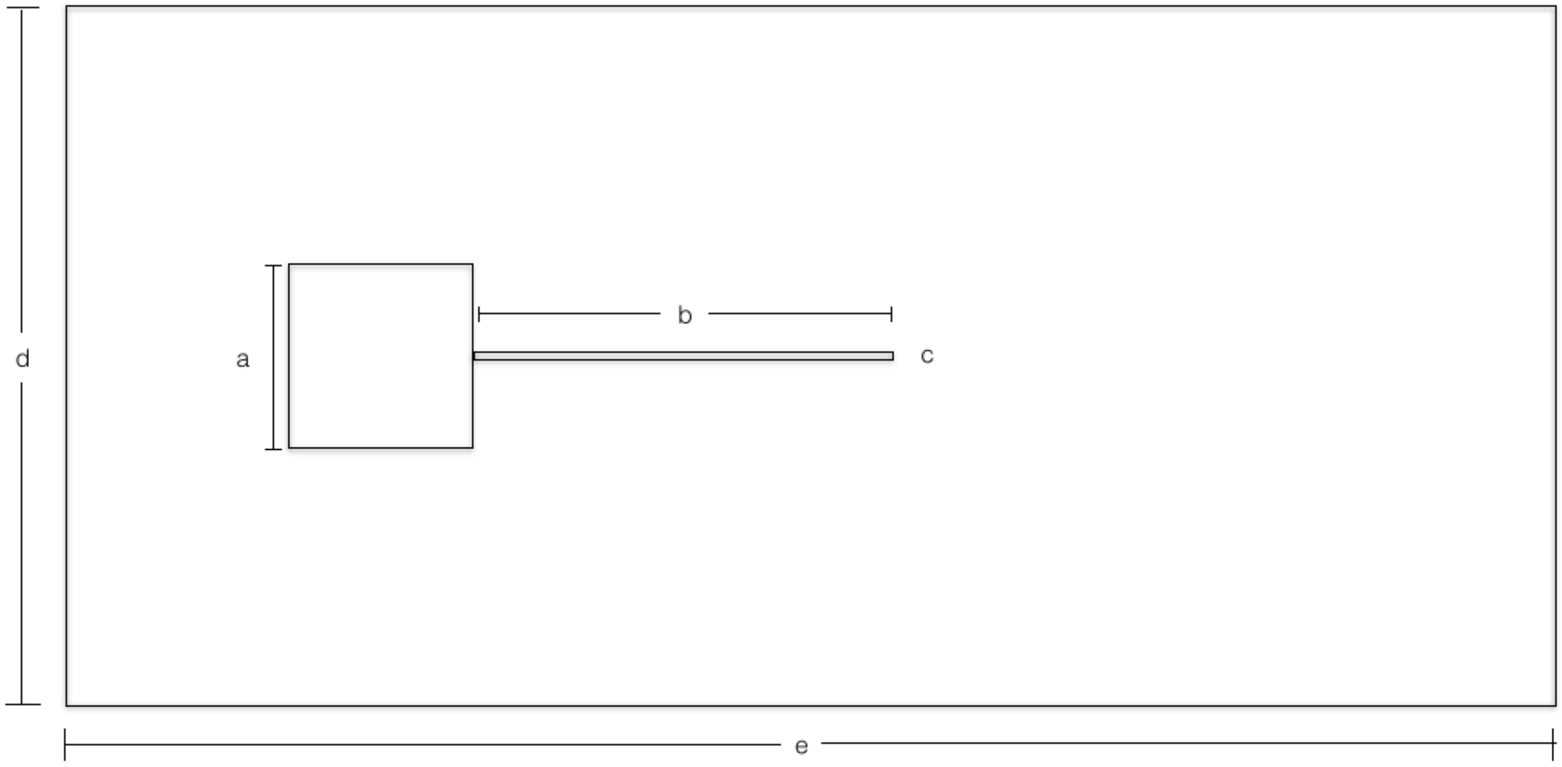
$\mathbf{t_s}$ is the traction on the interface in the solid domain

The Added Mass Effect

- Relevant for incompressible fluids
- Physical instability that destroys the convergence of simple coupling algorithms
- Appears when the fluid's and solid's density is similar
- Goes worse when time step decreases

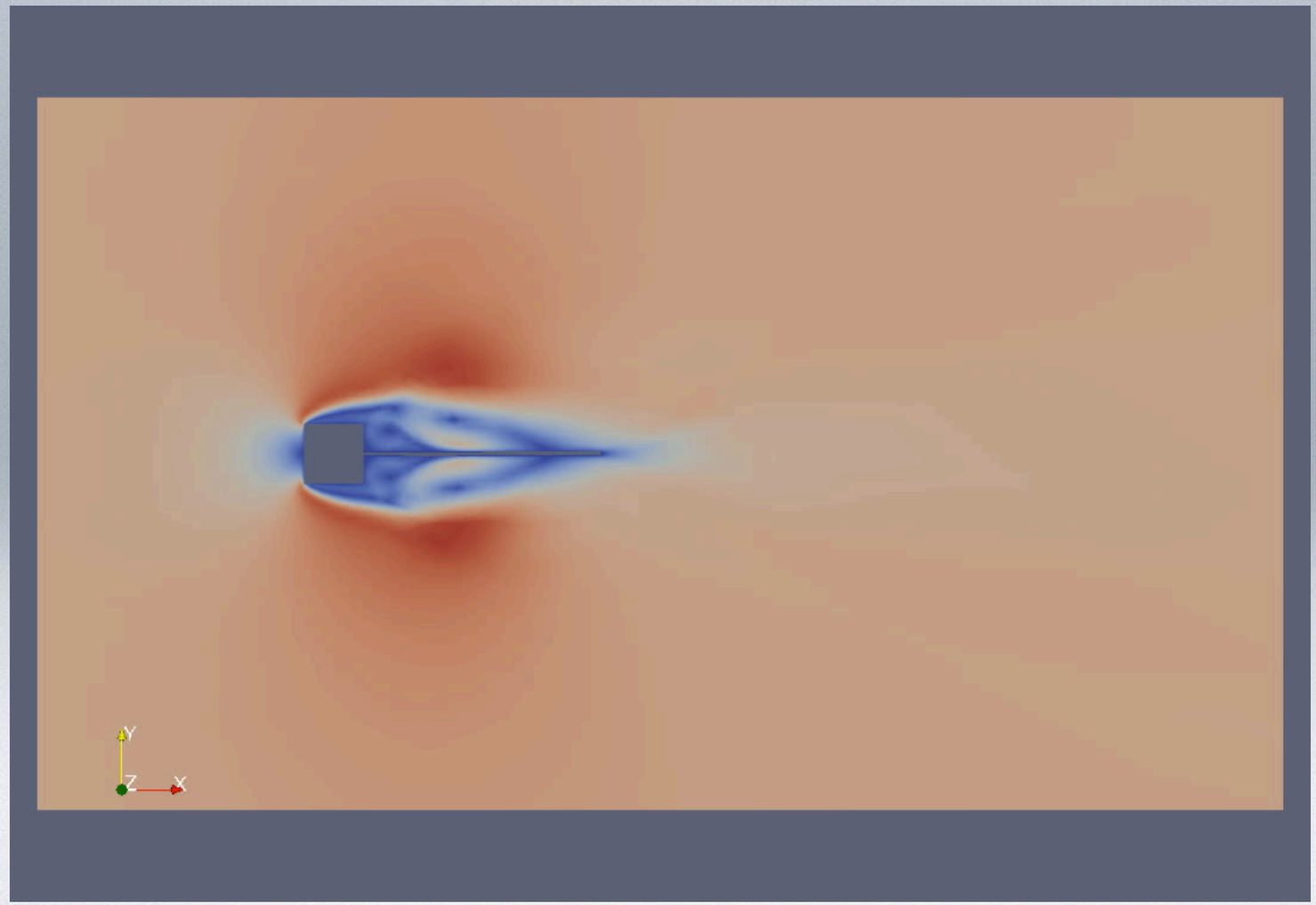
Test case

Inflow

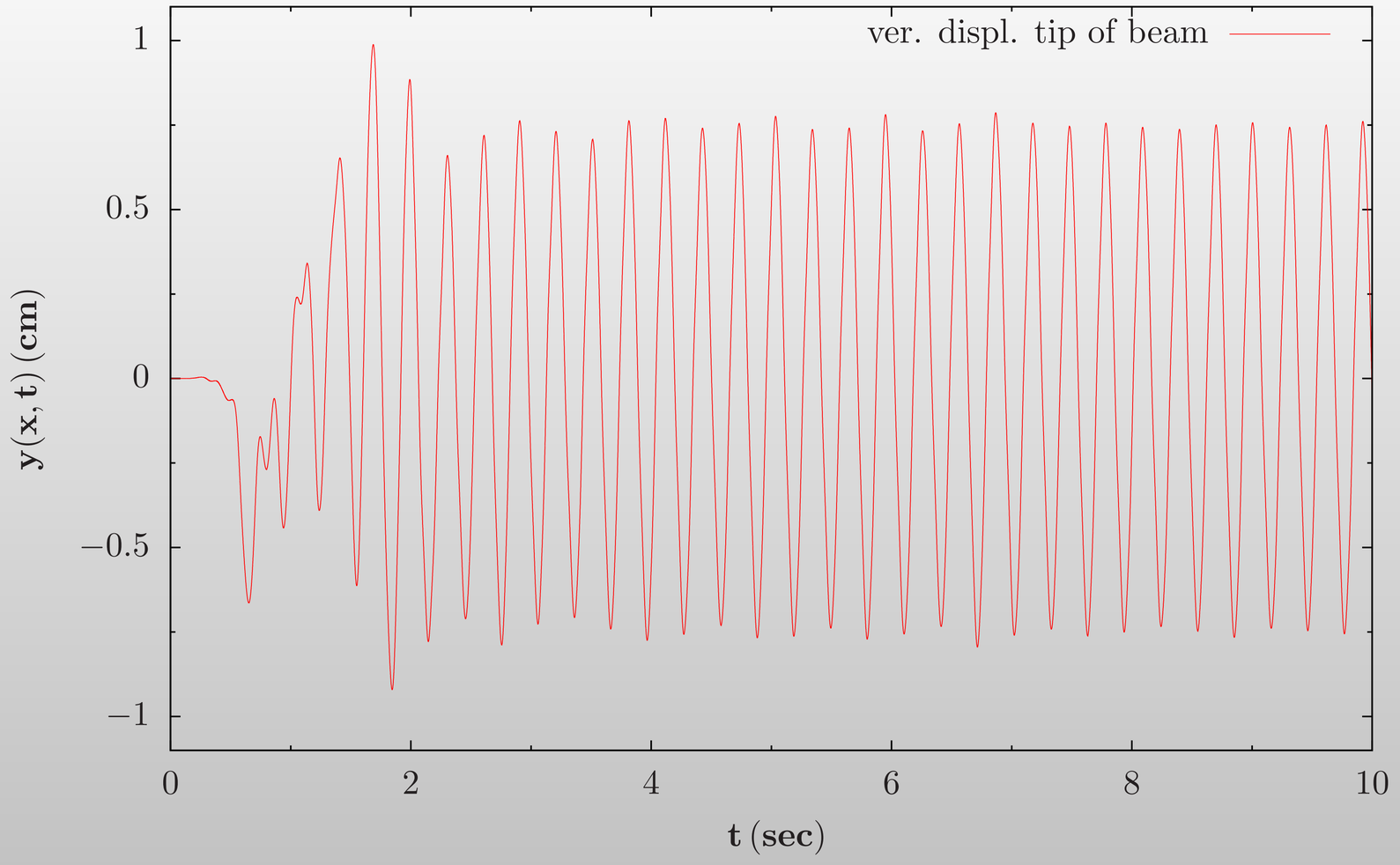


Outflow

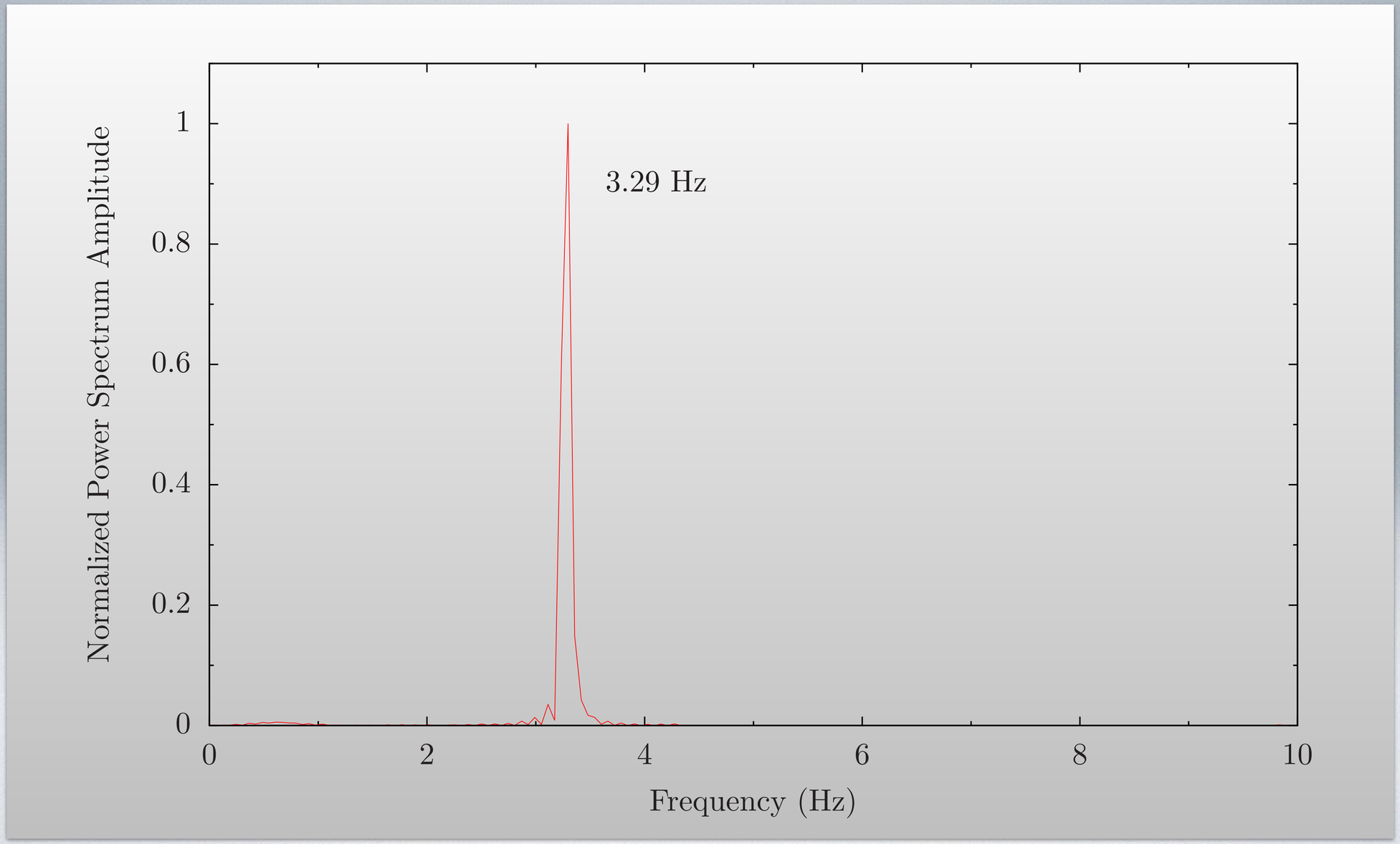
FSI problem results



FSI problem results

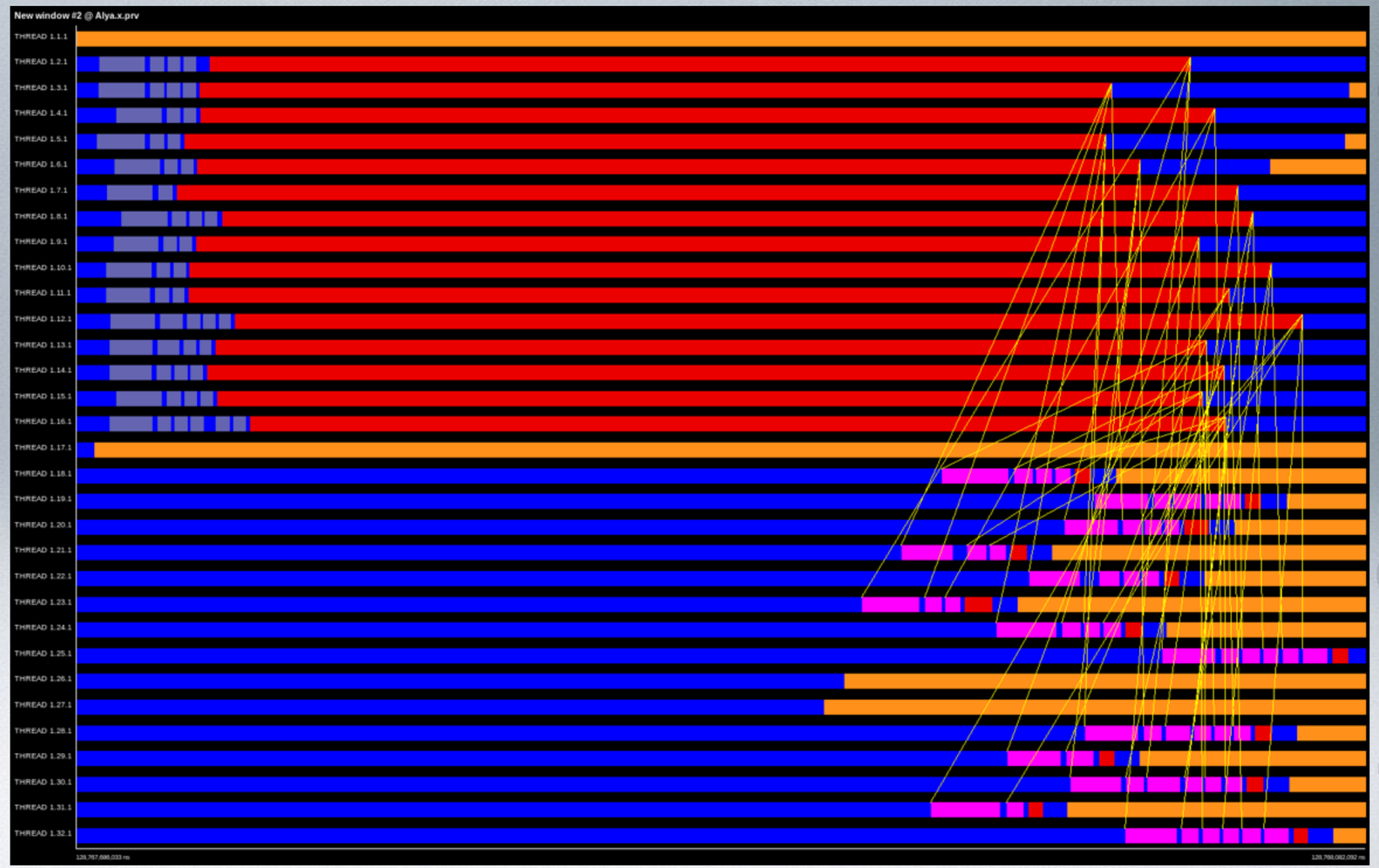


FSI problem results

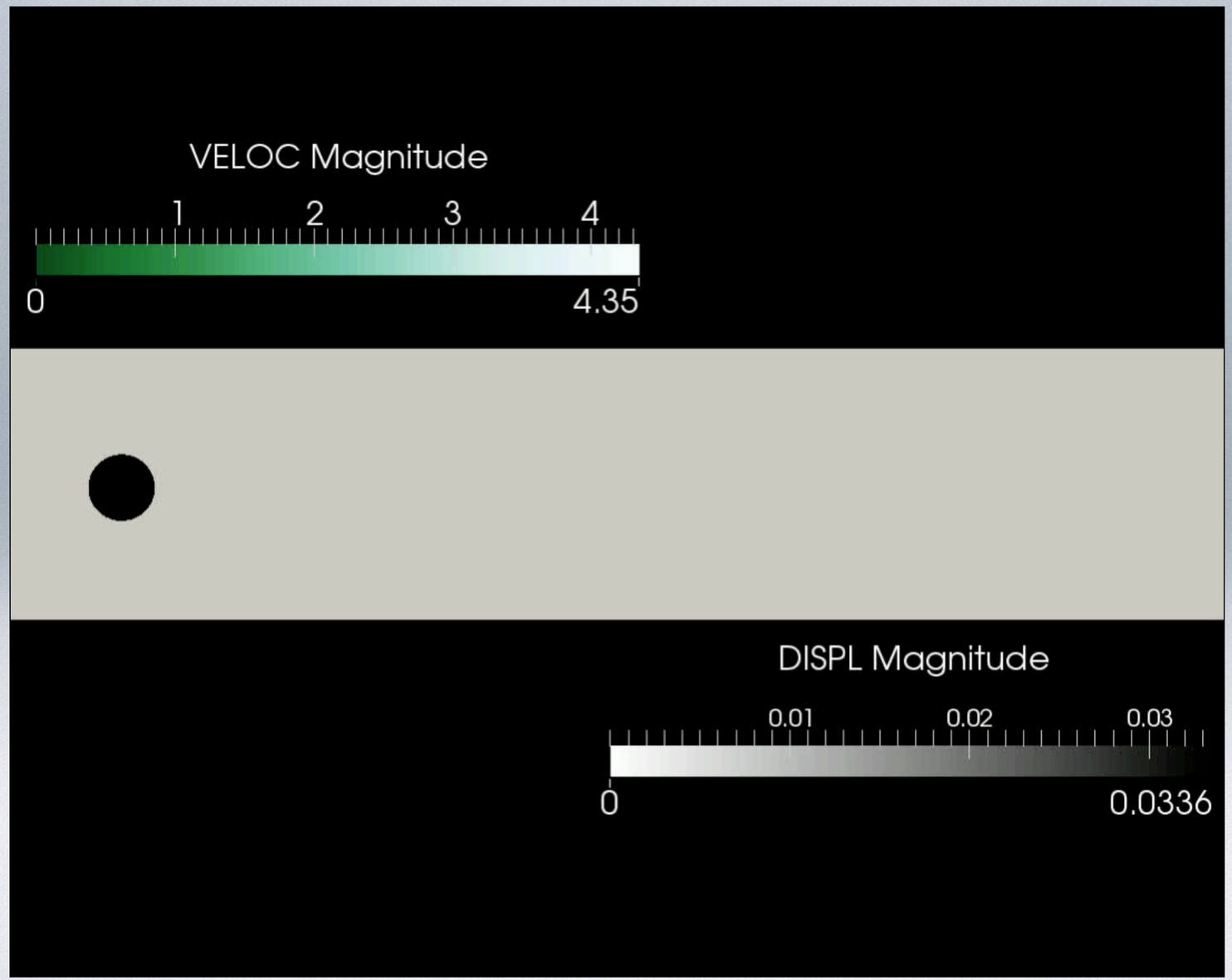


FSI problem results

- Calculating
- Idle
- Receive
- Send
- Global communication



FSI problem results



FSI problem results

Benchmark challenge:

Fluid side

1 672 326 elements in 32 MPI processes

Constant inlet velocity (to be changed)

Free outflow

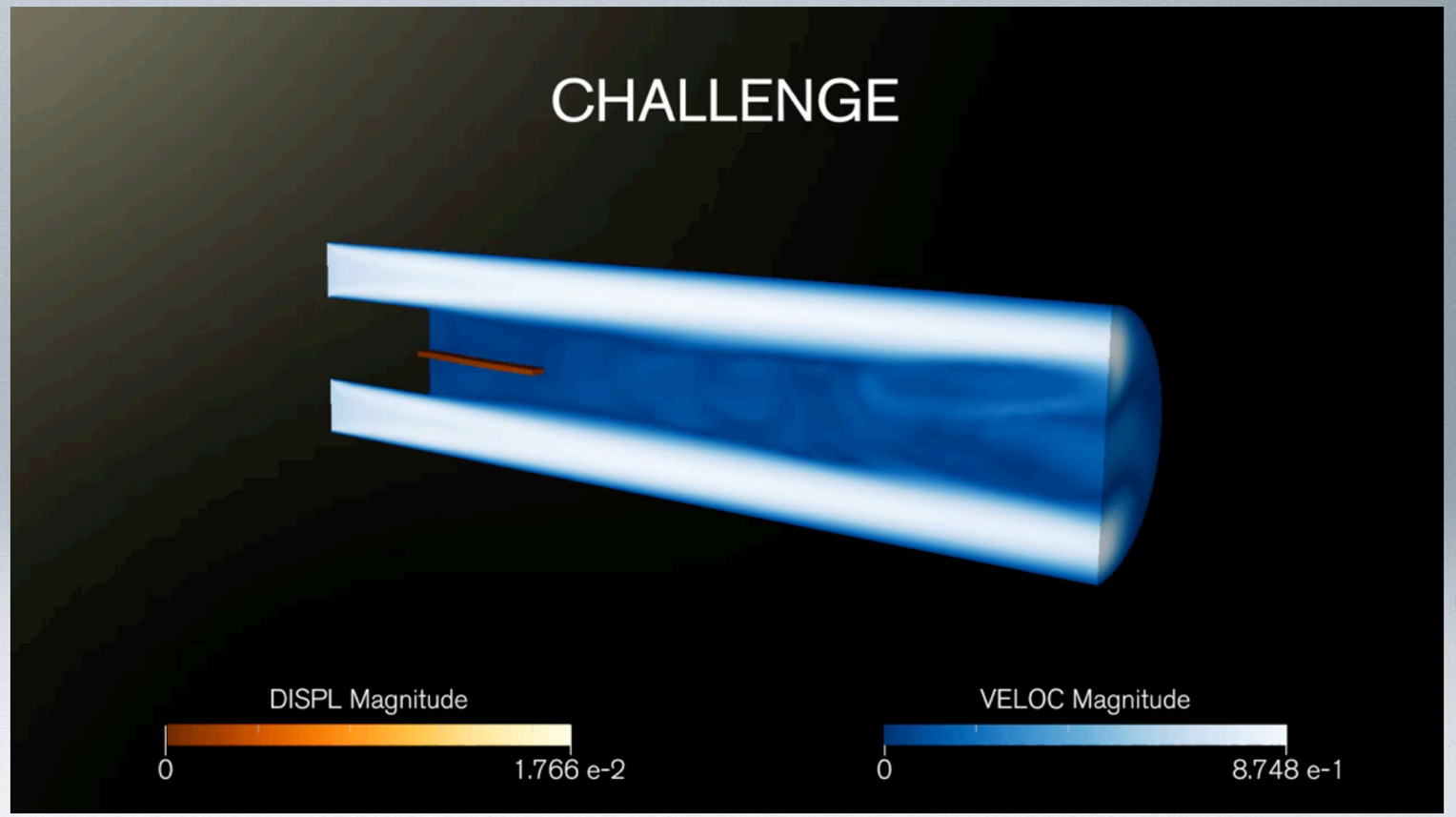
Solid side

19 200 elements in 15 MPI processes

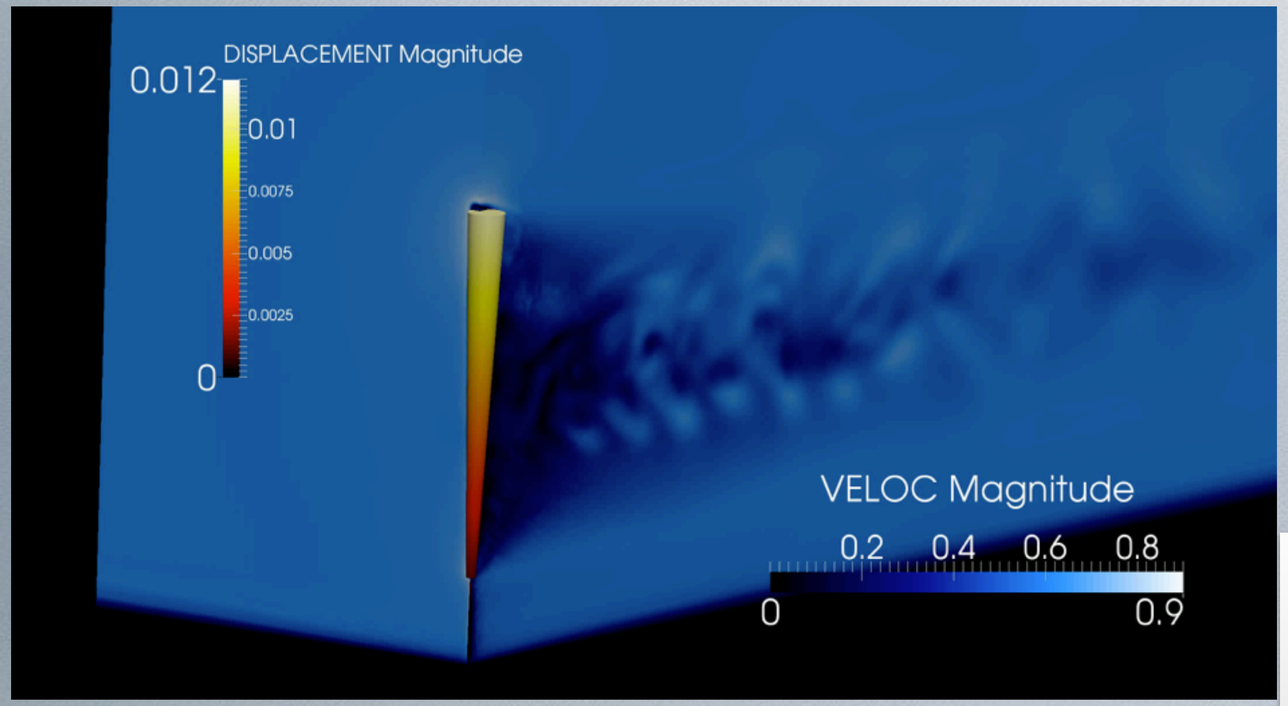
Tuned with iso-linear model (to be changed)

FSI problem results

- Constant plane velocity inlet profile
- Iso-linear model for solid beam
- Maximum displacement of the beam ~4.63mm

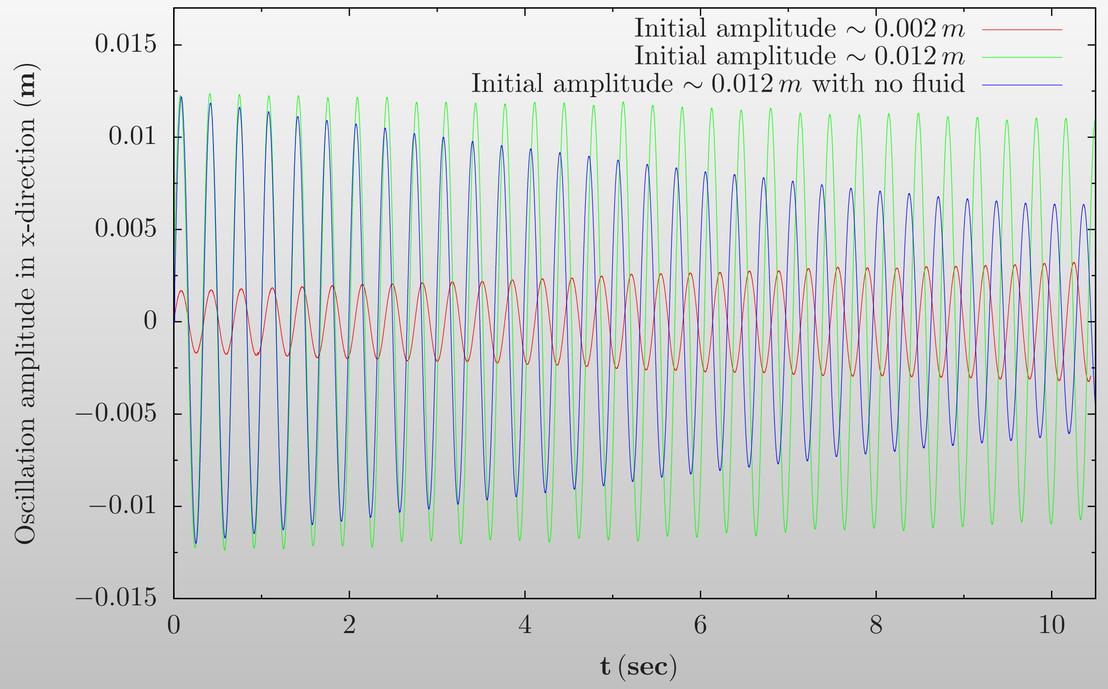


Vortex-Bladeless



- Device of 54 cm height, maximum diameter of 2 cm
- Wind velocity of 0.61m/s (Re~500)
- Simulation done with the Alya system

- 3,558,287 elements for the fluid
- 442,932 elements for the solid
- 357 processors for fluid code
- 129 processors for solid code
- 100,000 cpu hours in MN3

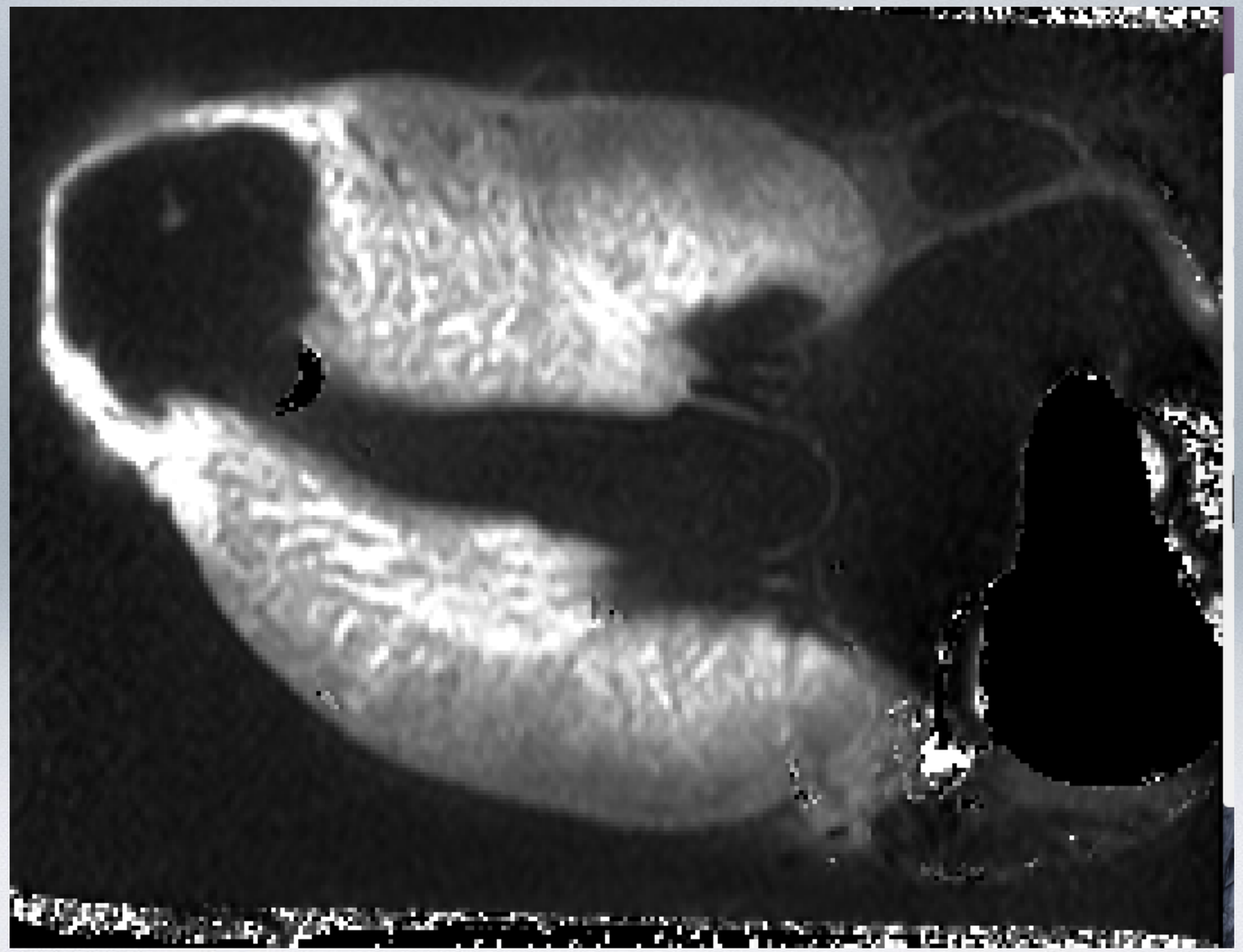


Parallel & Asynchronous Coupling of Fluid Structure Interaction problems in large vessels

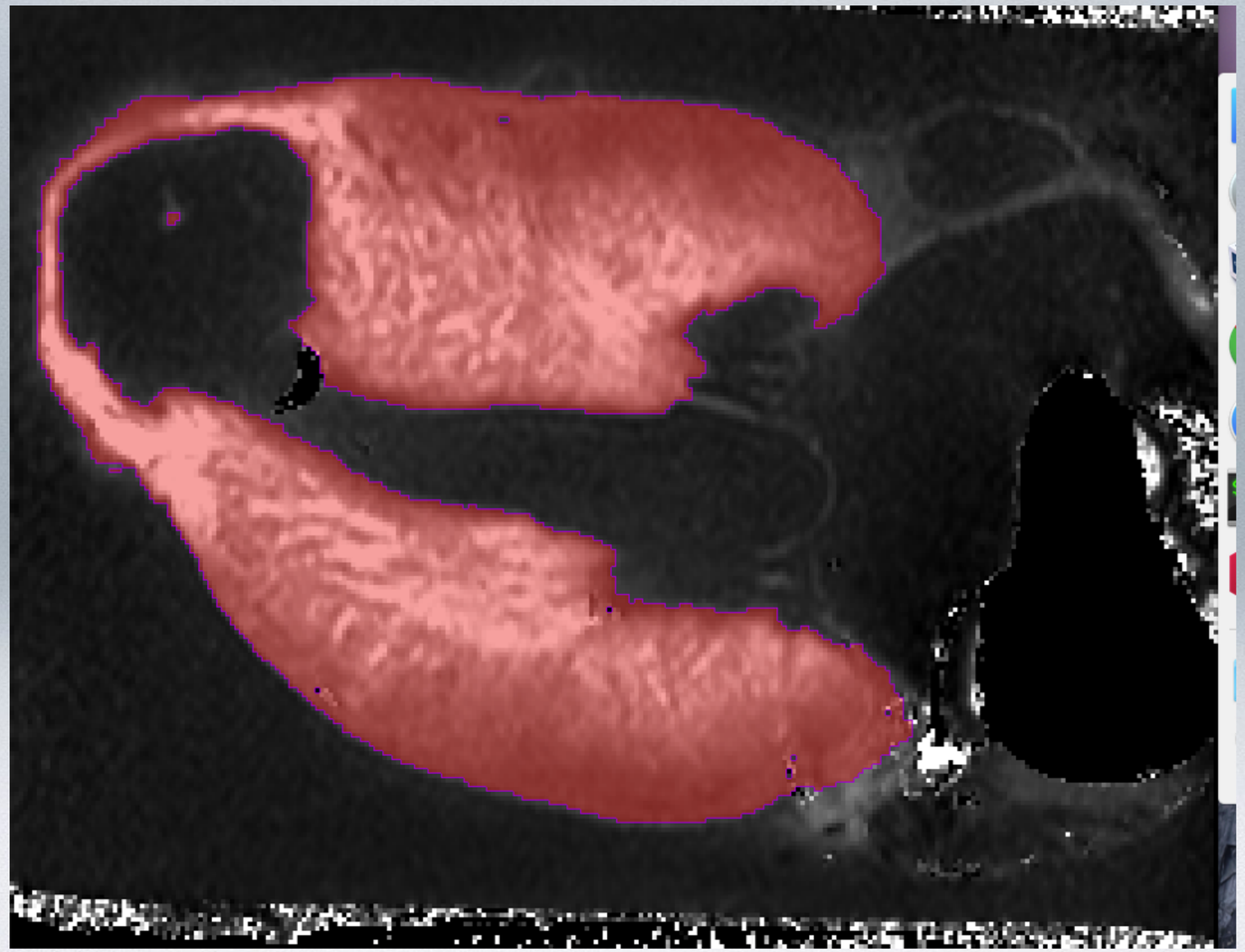
Particular problems of FSI in biological systems

- Complex geometry
- Non-automatic segmentation
- Boundary conditions?
- Coupling with physiological models
- All FSI problems

Obtaining computational domain from medical images



Obtaining computational domain from medical images



Boundary conditions

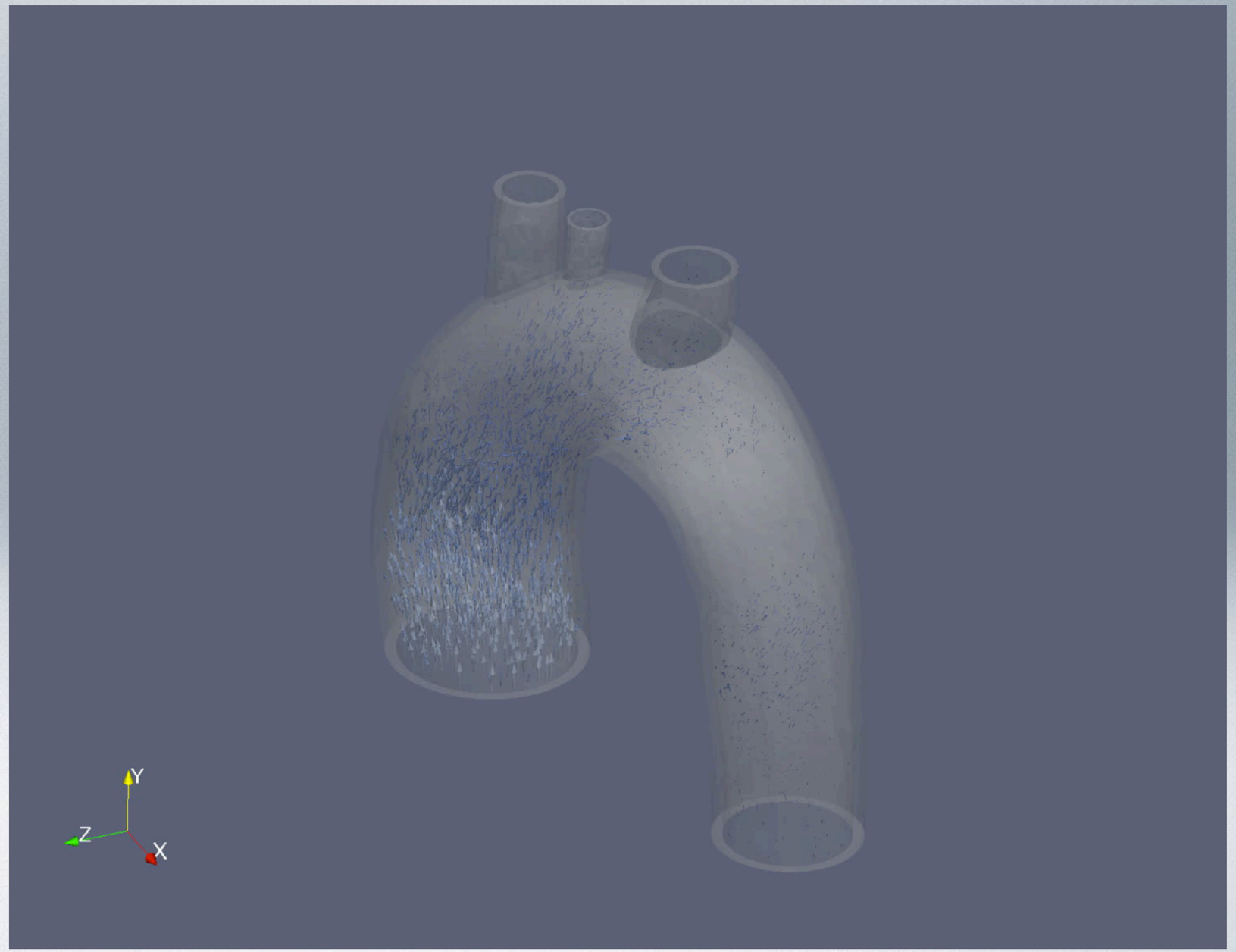
- Free outflow
- Windkessel model
- 4D Ultrasound measurements

FSI problem results

Collaboration with:

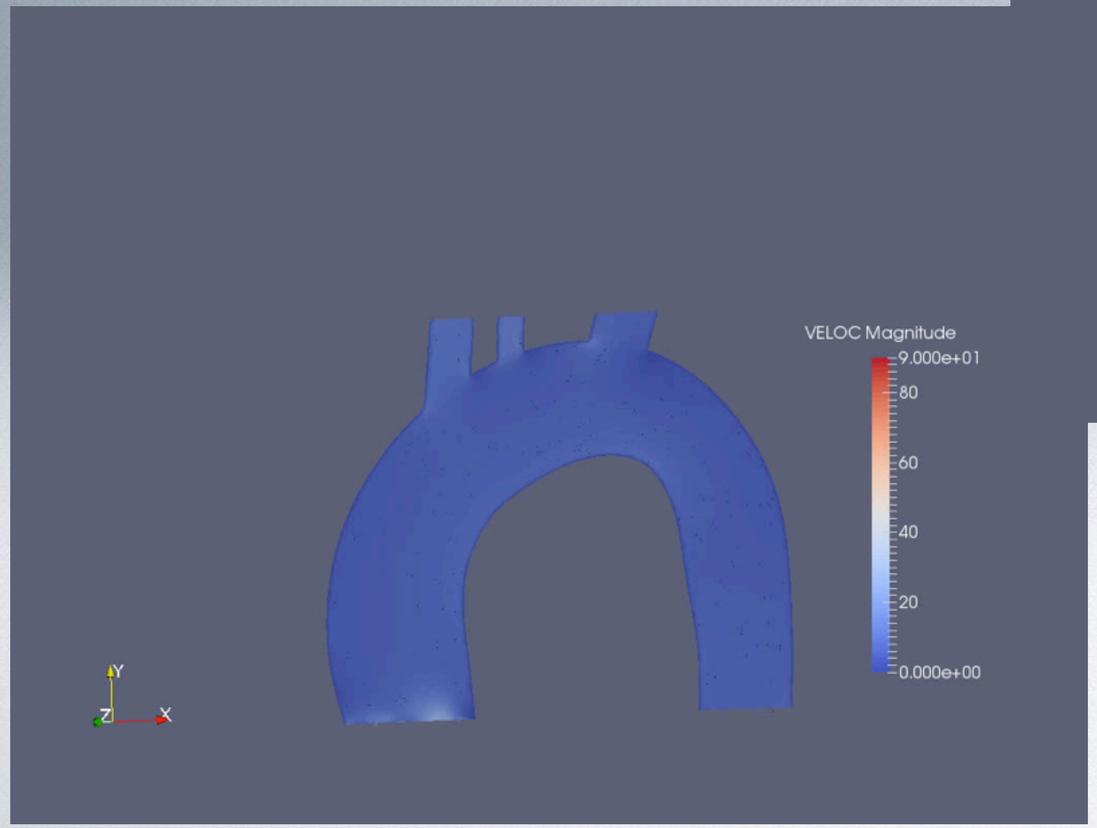
Ramon Pons
Jordi Martorell

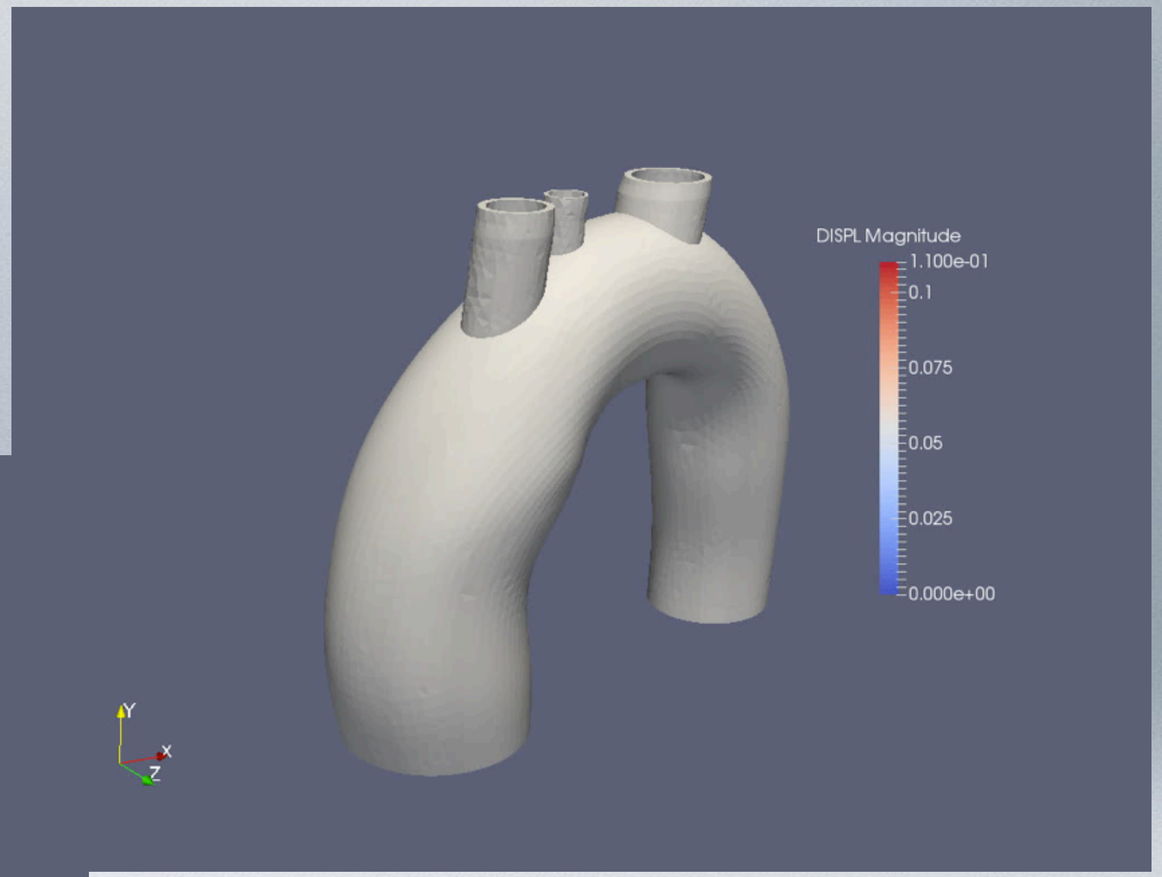
IQS, Barcelona, Spain



FSI problem results

Geometry and velocity coming from medical measurements



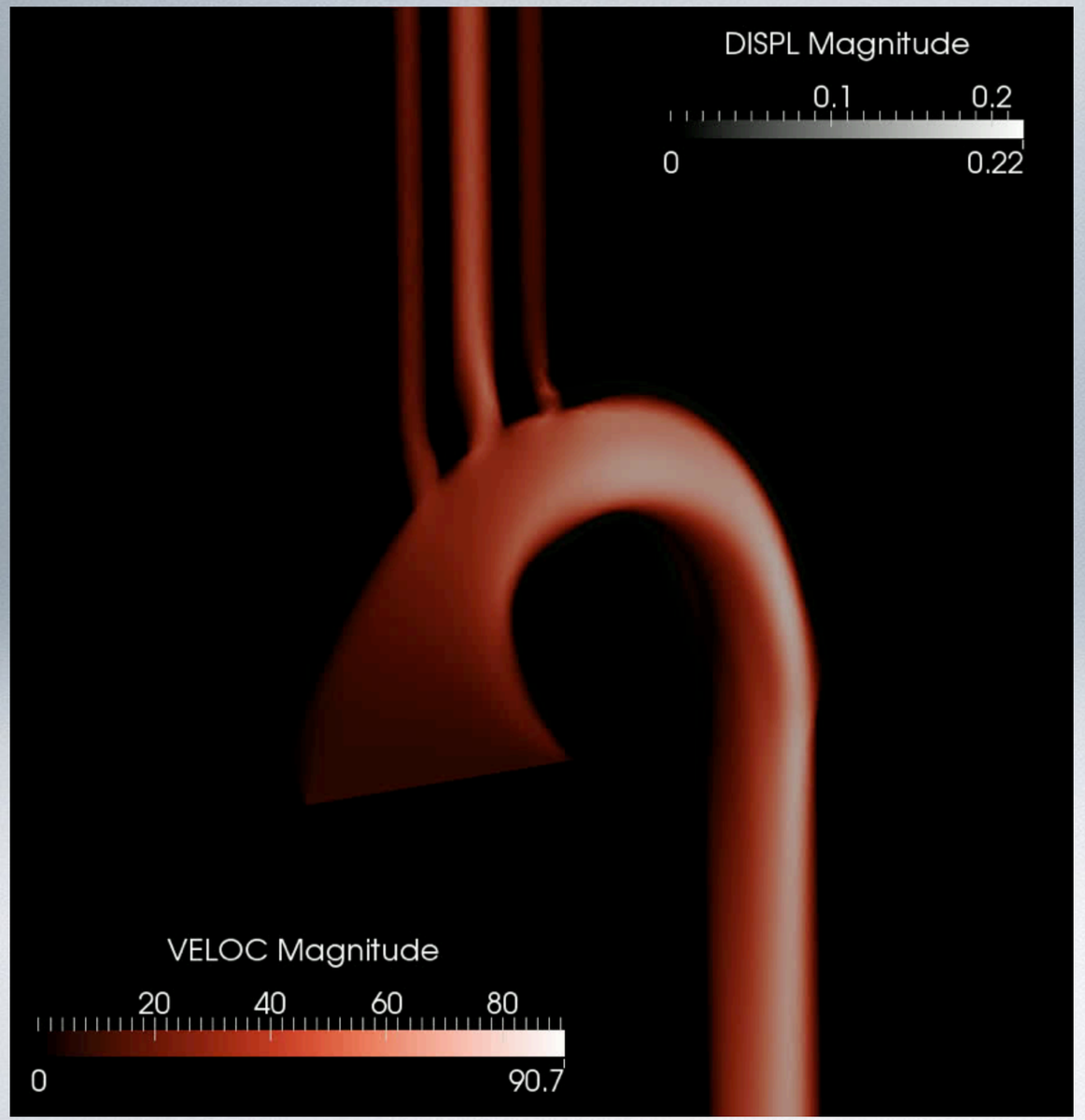


Collaboration with:

Ramon Pons
Jordi Martorell

IQS, Barcelona, Spain

FSI problem results



Geometry coming from medical measurements

Collaboration with:

Ramon Pons
Jordi Martorell

IQS, Barcelona, Spain

Conclusions

Multi-physics problems are exciting and defying (let’s have fun)

Multi-code coupling poses new computational challenges, and gives place to new and elegant solutions enhancing collaboration

Applications of HPC FSI problems are relevant to industrial and real life research fields

juan.cajas@bsc.es